INCLUYE CD-ROM

LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SÓLO PROGRAMADORES

Precio: 6 € (España) (IVA incluido) • AÑO XIV. 2.ª ÉPOCA • Nº 160 • UNA PUBLICACIÓN DE: REVISTAS PROFESIONALES S.L.



De regalo
el nº 158 de
Solo Programadores
en formato pdf

JAVAHISPANO
Actualidad Java

DISPOSITIVOS MÓVILES
Primeros pasos con Android (y III)

Construcción de aplicaciones con J2ME Polish (II)

DISEÑO
Programando en Java la Web Semántica con Jena (III)

MIDDLEWARE
JavaCup 2008, segunda edición torneo de fútbol virtual Java

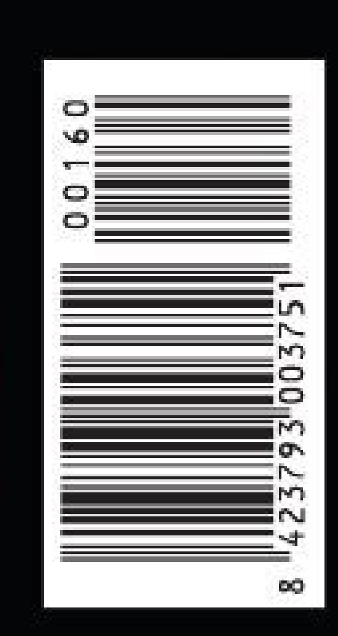
Hibernate y la sencillez de la capa de persistencia en JAVA

Java Media Framework (y II)

VÍDEO-TUTORIAL
Disco duro remoto

RSS con Java

La generación de RSS se ha convertido en una tarea común en el desarrollado de aplicaciones Web. En esta serie se estudiará cómo hacerlo con Java, utilizando su API estándar, de una forma robusta y eficiente.



Noticias, Lecturas, Actualidad y CD-ROM



www.jovenesenred.es/hostinet
únete al plan

Y Tú, ¿eres un Joven en Red?







planavanza





Y si ya no te dejan ser joven en red, nosotros te ofrecemos:

PLANES DE HOSTING
1'95 €/mes*
110 MB de disco duro
1 GB de transferencia
Correos ilimitados
Bases de datos MySQL

DOMINIOS .es
a 4'95 €

PACK MULTIDOMINIO
19'95 €/mes*
500 MB de disco duro
10 GB de transferencia
Correos ilimitados
Bases de datos MySQL

Más información en: 902 011 590 | info@hostinet.com

* Precio variable dependiendo del número de dominios que se posean. Para más información, consultar tabla de precios en nuestra página web, **www.hostinet.es**





Número 160

Edita: REVISTAS PROFESIONALES S.L.

solop@revistasprofesionales.com

C/Valentin Beato 42, 3ª 28037 - Madrid

www.revistasprofesionales.com

Editor

Agustín Buelta

Coordinación Técnica-Redacción

Ricardo Álvarez

Colaboradores

Abraham Otero, Juan Martos, Jorge Rubira, Guillem Alsina, Adolfo Aladro, Aitor Almeida, David Sainz, David Roldán, Diego López Nicolás Velasquez

Maquetación

Raúl Clavijo - Alfonso Sabán

Departamento de Publicidad

Felipe Ribagorda

Tel.: 91 304 87 64

Delegación en Barcelona

C/ Rocafort, 241/243, 5º 1ª

Mariano Sánchez

Tel.: 93 322 12 38

Dpto. Suscripciones

Tel: 91 304 87 64

Fax: 91 327 13 03

Impresión

L.M.S. Solución Gráfica

ideasimpresion@telefonica.net

Distribución



Saturnino Calleja, 7
28002 Madrid
Tfno. 915 864 933

DISTRIBUCION EN MEXICO
DIMSA - C/ Mariano Escobedo, 218
Col. Anáhuac. 11320 México, D.F.

DISTRIBUCION EN ARGENTINA
Capital Federal: Distrimachisa
Interior: York Agencysa - Tlf: (5411) 433 150 51

Depósito legal: M-26827-1994

PRINTED IN SPAIN

P.V.P. 6 euros

EDITORIAL

Se dice que "lo prometido es deuda". Pues bien, tal como anunciaba en el número anterior mi intención de mejorar en lo posible la revista, en esta ocasión además de los contenidos habituales del CD, incorporamos un documento en formato pdf con el nº 158 de Solo Programadores completo.

De esta forma, los lectores habituales podrán ir coleccionando la revista de una forma cómoda y liberando de peso esa estantería que comienza a tomar cierta curvatura.

Aunque en este tipo de materias que tratamos en la revista, su vigencia es mínima, por el mismo motivo anteriormente expuesto, estamos estudiando la posibilidad de en algún número relevante por su fecha (vacaciones de verano, Navidad, etc.), regalar en el CD habitual, 1 año completo de la revista.

Deseamos que este tipo de iniciativas sean del agrado del Lector y, como siempre, estamos abiertos a las sugerencias y críticas que deseen enviarnos.

SUMARIO



Microsoft continuará suministrando Windows XP para las computadoras ultraportátiles de bajo coste



Esta versión del penúltimo sistema operativo de Microsoft se encontrará a disposición de fabricantes e integradores hasta Junio de 2010, un año después de que deje de instalarse en el resto de las computadoras portátiles y de sobremesa.

El ya veterano sistema operativo de la compañía de Redmond podría vivir una segunda juventud gracias a las computadoras ultraportátiles baratas cómo el XO del proyecto OLPC o el Eee PC de Asus, que tan de moda se están poniendo en la actualidad.

El movimiento del gigante de Redmond persigue destronar a Linux como el "rey" de éste tipo de máquinas, ya que hasta ahora el sistema del pingüino ha equipado a la mayoría de iniciativas de máquinas ultraportátiles de bajo coste cómo los antes mencionados XO y Eee, aunque ninguno de los dos ha cerrado la puerta al sistema de Microsoft: el primero mediante un acuerdo llevado a cabo personalmente por el fundador del proyecto OLPC, Nicholas Negroponte, y el ex-presidente de Microsoft, Bill Gates, a quien les une una amistad personal de varios años. En el segundo caso, la máquina de Asus incorpora un DVD con los drivers y aplicaciones necesarias para que Windows XP pueda instalarse y funcionar correctamente.

Linux es una solución ideal para reducir el coste de estas máquinas, que es precisamente lo que la filosofía de las mismas (ofrecer una gran portabilidad aunque sea a costa de prestaciones reducidas por un precio realmente módico) obliga a hacer. Además, el amplio abanico de opciones para configurar una máquina cómo los diversos gestores de ventanas y entornos de escritorio que van desde los más ricos y que consumen más recursos hasta los más sencillos pero ligeros, permiten al fabricante la flexibilidad de adaptar a su creación un sistema realizado cómo un traje a medida.

Así pues, la compañía de Redmond no quiere ver el futuro de su actual dominio en el segmento de los sistemas operativos comprometido por la expansión de Linux en una clase de máquinas de la que se prevé una gran explosión a corto/medio plazo, con ventas no solamente para países del llamado "tercer mundo" (con menor poder adquisitivo; tercer mundo tiene una connotación despectiva propia de un término acuñado en los países occidentales), sino también en lugares cómo Europa Occidental o Norteamérica.

Muchos usuarios utilizan y utilizarán las computadoras ultraportátiles cómo complemento a sus máquinas desktop, sincronizando ambos dispositivos para intercambiar información, por lo que si se acostumbran a Linux en la ultraportátil es posible que también sientan la tentación de cambiar al sistema del pingüino en su sobremesa o portátil más potente. La misión de Microsoft es, obviamente, impedir que se dé esta situación.

Concretamente, la versión que se pondrá a disposición de fabricantes y usuarios será Windows XP Home, la más simple (por lo menos en teoría) en cuanto a gestión por parte del usuario final, y estará disponible hasta el 30 de Junio de 2010, un año después de que haya dejado de estarlo para computadoras de sobremesa y portátiles más potentes.

Windows XP Home estará al alcance de los fabricantes e integradores de ultraportátiles en formato OEM, para instalación en máquinas nuevas. Microsoft ha publicado en su sitio web una sección[1] dedicada a ofrecer el soporte e información necesaria para el funcionamiento de XP Home en éste tipo de equipos.

También se ha dejado claro que esta será la única versión de Windows XP que sobreviva a la "muerte" de éste sistema operativo, cuya fabricación se suspenderá en Junio de éste año y su suministro a los integradores un año más tarde. Microsoft quiere potenciar a Windows Vista pese a que ha tenido que aumentar el periodo de vida de XP debido a la demanda del mercado, y a que el éxito del nuevo sistema se basa sobretodo en las ventas de nuevas computadoras.

Más información:

Nota de prensa de Microsofthttp://www.microsoft.com/presspass/features/2008/apr08/04-03xpeos.mspx

[1] http://www.microsoft.com/unlimitedpotential/ULPC.mspx



Publicada la versión 2.5 de WordPress

Gran número de mejoras significativas para la nueva versión de uno de los sistemas de blogs más populares y utilizados en Internet. WordPress[1] es un CMS (siglas que corresponden a Content Management System, Sistema de Gestión de Contenidos en inglés) muy utilizado por todo tipo de bloggers y creadores de páginas web. De hecho, y junto a Movable Type[2], es una de las soluciones de publicación de contenidos para Internet más populares y utilizadas hoy en día, y muchos proveedores de servicios de Internet proporcionan a sus clientes un hosting con servicio de blogs mediante WordPress incluido.

Éste software es desarrollado en PHP y bajo licencia GPL por una comunidad independiente de programadores. La nueva versión 2.5 recoge las sugerencias que los usuarios han realizado durante estos últimos meses. El cambio más notable lo ha registrado la interfaz de usuario, hasta tal punto que los mismos responsables del software hablan de la remodelación más importante desde la versión 1.5 .

El Dashboard, la página principal que vemos cuando entramos en el gestor, se construye a partir de ahora mediante una serie de Widgets que nos muestran información de utilidad sobre el sitio web, los posts, o los enlaces que nos hacen desde otras páginas. Esta página de inicio puede personalizarse a gusto del usuario, de forma que por ejemplo podemos incluir las noticias de un periódico para que cuando entremos en el sitio las veamos directamente.

Al ser los blogs un fenómeno que ha posibilitado utilizar los gestores de contenidos específicos de estas herramientas para todo tipo de publicaciones, nos encontramos con muchos sitios web que son realizados por un colectivo de personas. En estos casos, más de una vez se ha dado la circunstancia que dos usuarios editan simultáneamente y sin saberlo el mismo post, con el resultado de que el último que lo guarda es quien en realidad salva los cambios, perdiéndose el trabajo (incluso de horas) que la otra persona ha volcado en su post. Para evitar estas situaciones, el nuevo WordPress bloquea el post de forma que nadie pueda sobrescribir el trabajo de otro realizando ediciones concurrentes.

El tratamiento de imágenes en el gestor también mejora. A partir de ahora es posible subir más de una imagen simultáneamente al sitio (lo que es extrapolable a cualquier tipo de archivo), y ordenarlas cómo galerías. También es muy fácil incluir una galería en nuestro post con solamente un tag.

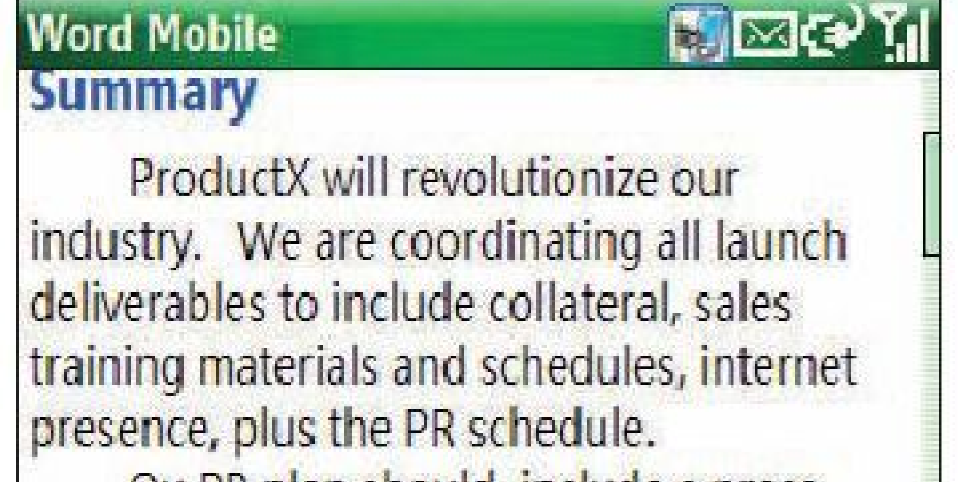


Otras novedades destacables de WordPress 2.5 son:

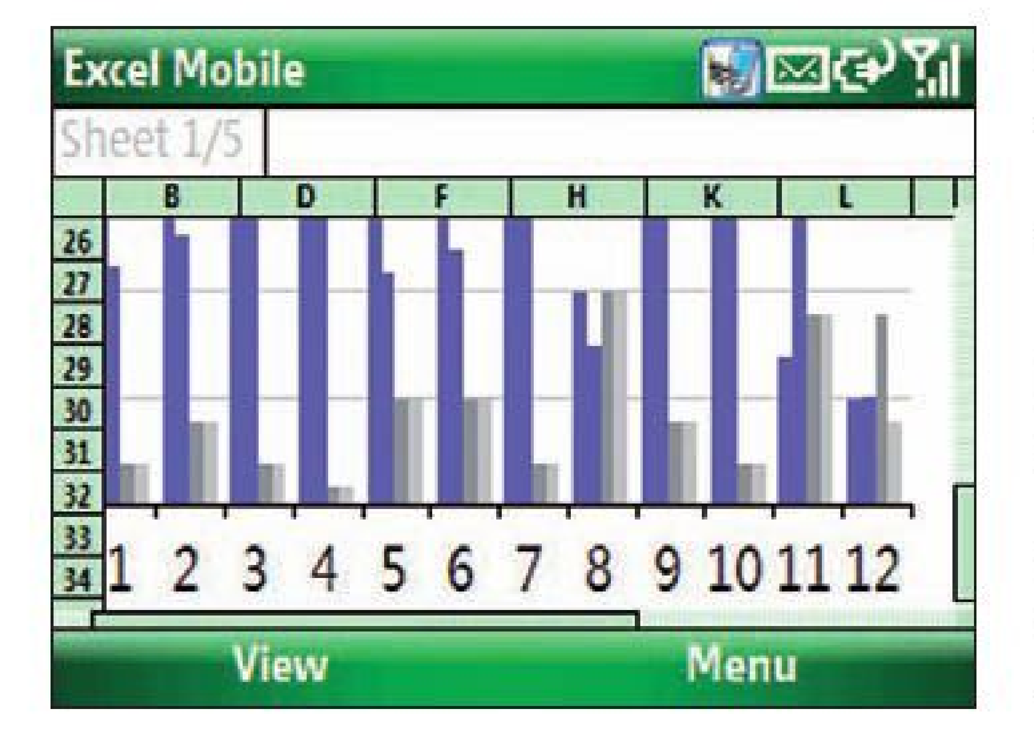
- Actualización automática de plug-ins, siempre y cuando estos consten en el directorio oficial de plug-ins.
- El editor de texto para los posts es el TinyMCE 3.0, con mejor integración en Safari
- Añadido soporte para ficheros JPEG con metadatos en formato EXIF. Los metadatos podrán ser utilizados en las entradas del gestor.
- La utilidad de búsqueda ahora cubre no solamente posts, sino también páginas.
- Añadido soporte para tags sin necesidad de utilizar plug-ins externos
- Mejoradas las APIs para desarrolladores.

Más información:
Anuncio oficial por parte de WordPress http://wordpress.org/development/2008/03/wordpress-25-brecker/
[1] http://wordpress.org/
[2] http://www.movabletype.org/

Microsoft quiere ofrecer un completo navegador web para dispositivos móviles



Soporte para multimedia y aplicaciones RIA acercarán al Internet Explorer móvil a la altura de Safari u Opera. La compañía de Redmond también presentó la versión 6.1 de su sistema operativo para dispositivos móviles. La compañía de Redmond quiere pisarle los talones a Apple que actualmente domina el panorama tecnológico de los navegadores web para dispositivos móviles con su versión de Safari. Pocket Internet Explorer está actualmente por detrás en tecnología y en popularidad de Safari y de Opera.



Desde los primeros dispositivos móviles con capacidad de navegar por Internet, los homólogos de los navegadores web para éste tipo de aparatos no han tenido las mismas posibilidades para renderizar las páginas web que visitan. Cómo consecuencia, la práctica mayoría de los sitios web se ven en nuestros dispositivos móviles distorsionados o incompletos.
La versión de Safari para el iPhone y el iPod Touch es el primer navegador móvil que renderiza las páginas web para que se vean de la misma forma que en un ordenador de sobremesa, soportando numerosos estándares hasta ahora fuera del alcance de los browsers móviles -al menos en la mayor parte de su implementación- cómo las hojas de estilo CSS. En la feria CTIA celebrada en Las Vegas (Estados Unidos) y en la que se exhibe buena parte de la industria de la telefonía móvil norteamericana y mundial, Microsoft ha dado a conocer[1] sus planes para una versión de Internet Explorer Mobile que estará a la altura de Safari y que permitirá disfrutar de la Web en todo su esplendor en nuestros teléfonos y PDA's.
En primer lugar, se podrán ver las páginas completas, lo que ya nos indica que dispondrá de una funcionalidad de zoom similar a la que ofrece el Safari móvil. La multimedia y las aplicaciones RIA son otros dos aspectos de los que hasta ahora la navegación móvil por Internet ha prescindido. Con el nuevo Internet Explorer Mobile, los internautas podrán acceder a sitios que utilicen Silverlight, Adobe Flash y ver vídeos H.264 .

Word Mobile

	Collateral	Sales Trai
Launch Date	100%	100%
D minus 1	95%	95%
D minus 5	90%	90%
D minus 10	85%	85%
D minus 30	75%	75%
D minus 60	35%	35%

View Menu

Se espera que el nuevo browser esté listo para enviar a los constructores de dispositivos móviles en algún momento del tercer trimestre de éste año, y que podamos ver el primer teléfono o PDA que lo utiliza a finales de 2008.

Windows Mobile 6.1

El anuncio del nuevo Internet Explorer Mobile ha eclipsado incluso al anuncio de la revisión del sistema operativo que le da vida, Windows Mobile 6.1[2] . En éste se han introducido cambios destinados a mejorar la experiencia de usuario en ambientes corporativos. Así pues, nos encontramos con mejoras en la gestión de los eventos desde la Home Page, ya que desde un sólo sitio podemos responder a recordatorios en nuestra agenda, llamadas perdidas u otros tipos de eventos. Esta centralización también se palpa en la configuración de las conexiones, ya sean estas WiFi, Bluetooth o GPRS.
Los teléfonos equipados con esta nueva revisión del sistema operativo para dispositivos móviles de Microsoft podrán verse en el mercado a partir del segundo trimestre de 2008.
[1]http://www.microsoft.com/presspass/press/2008/apr08/04-01WM61PR.mspx
[2]http://www.microsoft.com/presspass/press/2008/apr08/04-01EnterpriseMobilePR.mspx

Primer ataque cracker con consecuencias físicas para las víctimas

Un grupo de crackers cambiaron el fondo de la página web de un foro sobre Epilepsia para que destellease rápidamente, provocando precisamente ataques epilépticos a algunos internautas habituales del foro que padecen la dolencia.
Según publica[1] la versión online de la revista Wired, un asalto cibernético a los foros de la Epilepsy Foundation[2] norteamericana provocó efectos físicos en los internautas usuarios que padecen epilepsia.
Esta es una enfermedad que se caracteriza por ataques de tipo nervioso que provocan espasmos y que son debidos a una actividad eléctrica anormal en el cerebro. Una de las causas de esta actividad anormal pueden ser los estímulos visuales que presentan cambios constantes, cómo una serie de destellos consecutivos. Eso es precisamente lo que hicieron los asaltantes cuando según el artículo de Wired, postearon diversas entradas en los foros[3] del sitio web de la Epilepsy Foundation con imágenes en formato GIF animado que mostraban rápidos destellos y cambios de color. Pero al día siguiente refinaron su técnica, empleando JavaScript para redirigir a los usuarios a otra página fuera del sitio de la Epilepsy Foundation, esta más trabajada y que igualmente mostraba animaciones más "potentes" para provocar ataques a los enfermos de epilepsia.
Se desconoce con seguridad la identidad de los atacantes, aunque según informa Wired se cree que han sido miembros del grupo autodenominado Anonymous, que hace unos meses declararon una guerra virtual contra la Iglesia de la Cienciología, en la cual sufrieron más de un revés cómo atacar los servidores equivocados. Esto les dio mala prensa y les dejó en ridículo ante la comunidad hacker mundial.
La Epilepsy Foundation corrigió el problema retirando temporalmente los foros y purgán-

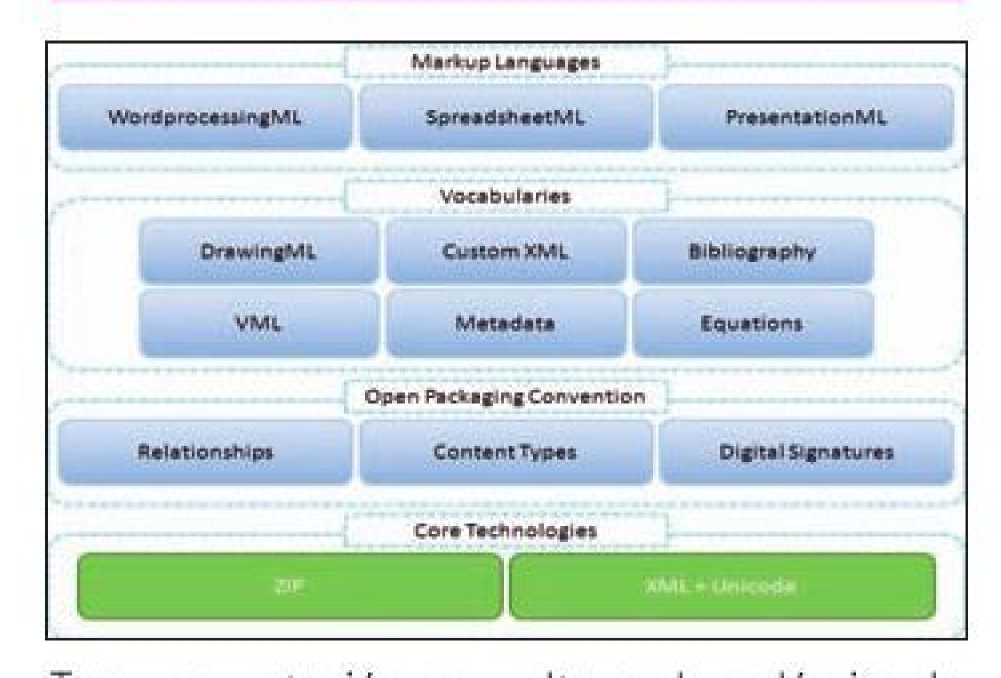


dolos del contenido dañino. Ahora ya vuelven a estar en línea.
No obstante, algunos habituales de éste foro sufrieron las consecuencias del ataque, aunque que se sepa, solamente se trató de efectos leves.

Crackers, no hackers

El uso del lenguaje importa, y más si se trata de conceptos controvertidos cómo el hacking y el cracking. Pese a que no hay un acuerdo unánime sobre el uso de los términos 'hacking' y 'cracking', se conviene en aceptar usualmente entre los medios especializados que mientras el hacker sigue un código ético de lo que considera aceptable o no para la sociedad, el cracker es el experto que ha cruzado la línea que separa lo legal de lo ilegal. Serían dos caras de la misma moneda, la 'buena' y la 'mala'.
No obstante, el vacío de acuerdo en el tratamiento del término hacker hace que muchas veces éste sea empleado cómo sinónimo de ciberdelincuente, especialmente en el contexto de los mass media. Éste es, curiosamente, el término empleado en el artículo de Wired.
[1]http://www.wired.com/politics/security/news/2008/03/epilepsy
[2] http://www.epilepsyfoundation.org/
[3]http://www.epilepsyfoundation.org/forums/index.cfm?nocookies=yes

Open XML ya es un estándar ISO



Tras una votación envuelta en la polémica, la compañía de Bill Gates se sale con la suya y ya puede exhibir orgullosa el sello de estándar dado por la ISO a su formato de documento para Office 2007.
Pese a toda la polémica generada alrededor de las votaciones, las acusaciones de presiones y sobornos que según algunos han hecho desde Microsoft a los miembros de distintos comités regionales de la ISO, la dura oposición realizada por un sector contrario a la aprobación que esgrime cómo bandera el formato OpenDocument, pese a todo esto y más, Microsoft ha conseguido que su propuesta para que el formato Open XML utilizado en Office 2007 sea aprobado cómo estándar ISO. Éste es un paso muy importante para la compañía de Redmond. La ISO (International Organization for Standardization) es una organización supra-nacional que marca una gran cantidad de estándares sobre diversos temas que se convierten prácticamente en ley, superando en muchos casos a los estándares nacionales. Casi todos los países del mundo tienen representantes en esta organización y acatan sus decisiones, tal vez más de lo que se puede decir de otras organizaciones en teoría más trascendentes cómo la ONU...
La propuesta de Microsoft ha sido polémica desde el principio, y ha generado una fuerte corriente opositora que tiene cómo principales integrantes a Google y a organizaciones relacionadas directa o indirectamente con el formato OpenDocument, que afirman que la especificación de formato presentada por Microsoft es opaca e incompleta, y acusan a la multinacional de Redmond de cometer varias irregularidades en el proceso para convencer a delegados regionales clave de que voten en sentido favorable.
Durante todo el proceso, Microsoft ha mantenido un silencio ejemplar, sin responder a las acusaciones presentadas de ninguna forma y refiriéndose solamente a la marcha del mismo proceso de aprobación. Cabe recordar que en primera instancia la propuesta del Open XML fue rechazada por un estrecho margen y devuelta a Microsoft con una serie de indicaciones de lo que debía ser mejorado y una segunda oportunidad, que ahora los chicos de Bill Gates han sabido aprovechar.
La resolución de la ISO se ha filtrado unos días antes que sea dada a conocer por la organización. Otros formatos de documento informático que han conseguido el rango de estándares oficiales son el PDF de Adobe, HTML u ODF (OpenDocument Format, utilizado por las suites OpenOffice.org y KOffice entre otras). Open XML era ya un estándar aprobado por la ECMA, aunque esta es una organización con mucha menos influencia y prestigio que la ISO.
Más información:
Nota de prensa de Microsoft http://www.microsoft.com/presspass/press/2008/apr08/04-01OpenXMLVotePR.mspx
Sitio de la comunidad Open XM http://www.openxmlcommunity.org/

Ya disponible la versión gratuita online de Photoshop

Permite retoques simples cómo la eliminación de los ojos rojos típicos del uso del flash de la cámara, y proporciona un espacio de almacenamiento online para crear una galería de imágenes. Su uso es gratuito previo proceso de alta.
Cumpliendo con lo anunciado hace ahora un año, Adobe ha puesto a disposición[1] de los internautas una versión gratuita online utilizable a través de un navegador web de su programa de retoque de imágenes Photoshop,



líder del mercado en éste sector. Por el momento se trata de una versión beta.
El nombre de esta aplicación de la Web 2.0 es Photoshop Express[2] y permite a los internautas archivar imágenes en línea (con un espacio de almacenamiento de hasta 2 gigabytes), ordenarlas, mostrarlas, compartirlas con los servicios más populares cómo Facebook, y realizar pequeños retoques. Al tratarse de un software online dependiente de un navegador web para su ejecución, no es tan potente cómo su "hermano mayor" para Windows y Mac OS, pero permite algunas funciones cómo el redimensionado de imágenes, la eliminación de los ojos rojos típica del flash, conversión de color a blanco y negro o distorsiones varias con el color o la propia imagen para realizar divertidos y vistosos efectos.
Todos los cambios se realizan de forma no destructiva, así que siempre contamos con la fotografía original para volver sobre nuestros pasos.
En el apartado de exhibición de imágenes nos encontramos con numerosos y espectaculares efectos para mostrar las imágenes y realizar la transición de una a otra, de manera que la galería de Photoshop Express puede ser utilizada para presentaciones de todo tipo.
Para utilizar Photoshop Express debemos darnos de alta en el sitio web del programa[2] siguiendo un proceso muy rápido en el que se nos solicitará el nombre de usuario, una contraseña y unos pocos datos personales más. A partir de aquí tendremos que esperar hasta 24 horas para recibir el mensaje de correo electrónico de confirmación y empezar a utilizar nuestra nueva cuenta de Photoshop Express.
Teóricamente, éste servicio es utilizable solamente para ciudadanos estadounidenses, pero no se han reportado problemas si uno se da de alta desde otro país y afirma ser ciudadano del país norteamericano.
Los requisitos para utilizarlo son disponer de un navegador web compatible con Flash 9 cómo pueden ser Internet Explorer, Firefox o Safari. Ya que utiliza la tecnología Flex, cuando accedamos al software éste comprobará si tenemos instalado el reproductor Flash 9 en nuestra máquina, y si no es así llevará a cabo el proceso de instalación. Éste componente es indispensable para que Photoshop Express funcione correctamente.



[1]http://www.adobe.com/aboutadobe/pressroom/pressreleases/200803/032708PhotoshopExpress.html
[2]https://www.photoshop.com/express

¿Puede el exceso de información de la Red crear adicción?

Basado en un caso real: un padre, temeroso de que su hijo se pasase demasiadas horas ante el ordenador viendo páginas pornográficas, jugando a videojuegos o navegando entre contenidos aún más controvertidos, decidió investigar. y descubrió que su hijo era adicto... a puntuar noticias en Digg.

Desde siempre se ha hablado de Internet cómo una autopista de la información, con miles de millones de datos referentes a cualquier aspecto de la humanidad. En resumen, con un grave problema de sobreabundancia de contenidos que también se ha puesto de manifiesto más de una vez. Cómo los "problemas" que puede causar la red a la sociedad.

Porque, desengañémonos: hay quien considera un problema las páginas pornográficas, otros la información para fabricar bombas o cometer atentados terroristas y otros la excesiva facilidad con la que se informa de cualquier hecho ya sea de una forma subjetiva u objetiva. En cualquier caso, Internet -como cualquier cosa en éste mundo- supone una amenaza para muchos y variados colectivos.

Uno de estos colectivos afectados es el de los padres con hijos menores de edad, cuyos sufrimientos basculan entre que sus retoños "surfeen" todo el día por páginas pornográficas o bien vean contenidos de tipo "antisocial" que los lleven a perpetrar un acto de violencia extrema cómo el vivido en la localidad norteamericana de Columbine o más recientemente en una pequeña localidad finlandesa. En ambos casos, los medios de comunicación generalistas destacaron el papel de Internet cómo un factor de riesgo para las influenciables mentes de los jóvenes.

Pero no todo son riesgos. En el sitio web Burbia se nos explica[1] el caso real de un padre preocupado por la actitud de su hijo respecto al ordenador por el gran número de horas que se pasaba ante la máquina, navegando por Internet. Y más después de una charla dada para los padres en la escuela local donde se les advertía de los "riesgos" de la Red de redes.

Tal fue la inquietud de éste abnegado progenitor que, violando la privacidad de su hijo, decidió investigar que es lo que éste hacía realmente en el ordenador, descubriendo gran cantidad de accesos a una página llamada Digg, que desconocía. Digg es lo que se ha dado en llamar un agregador social, un sitio al cual y después de pasar por un proceso de alta, pueden enviarse referencias sobre noticias diversas para compartir con el resto de usuarios y que estos las valoren. Una especie de "press clipping" (resumen de prensa) pero realizado con esfuerzo voluntario.

El chico (de 16 años de edad) se pasaba buena parte del horario extraescolar leyendo, puntuando y comentando noticias e informaciones de toda clase, desde divertidas sátiras políticas hasta artículos sobre física cuántica. Ahora es donde entra en juego la pregunta ¿a partir de qué punto deja de ser interés cultural para convertirse en adicción? aunque también ¿qué es exactamente lo que definimos cómo adicción? y lo que es aún más importante: ¿porqué los medios de comunicación de corte generalista sacan casi siempre a relucir el lado más oscuro de Internet sin explicar sus pros además de sus contras?

Formas de adicción hay muchas, y tal vez deberíamos considerar si ver todos los resúmenes de fútbol durante el fin de semana y los primeros días laborales es otro tipo de adicción, por poner un ejemplo, y no juzgar todo un medio de comunicación a la ligera por una parte oscura que tener, la tiene. Cómo todo en esta vida.

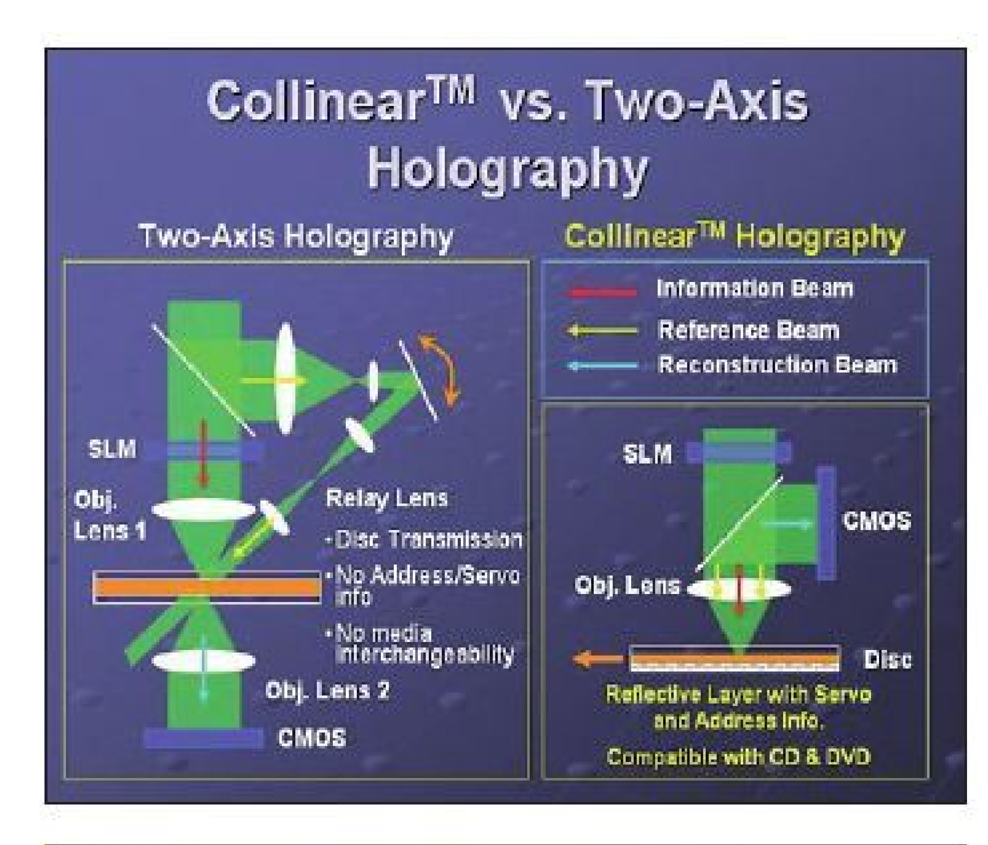
Pero lo que es aún más extraño: ¿puede haber adicciones "positivas"? La cantidad de información de todo tipo, banal pero también importante, que ha pasado y pasará ante los ojos de éste chico norteamericano protagonista de tan curiosa "adicción" hará de él una persona informada y culta (¿o no es la cultura un acopio de información?) y, si tiene un buen sentido común, sabrá procesarla, lo que será sin lugar a dudas en su provecho. ¿Cómo tomamos en consideración estos efectos beneficiosos de una presunta adicción?

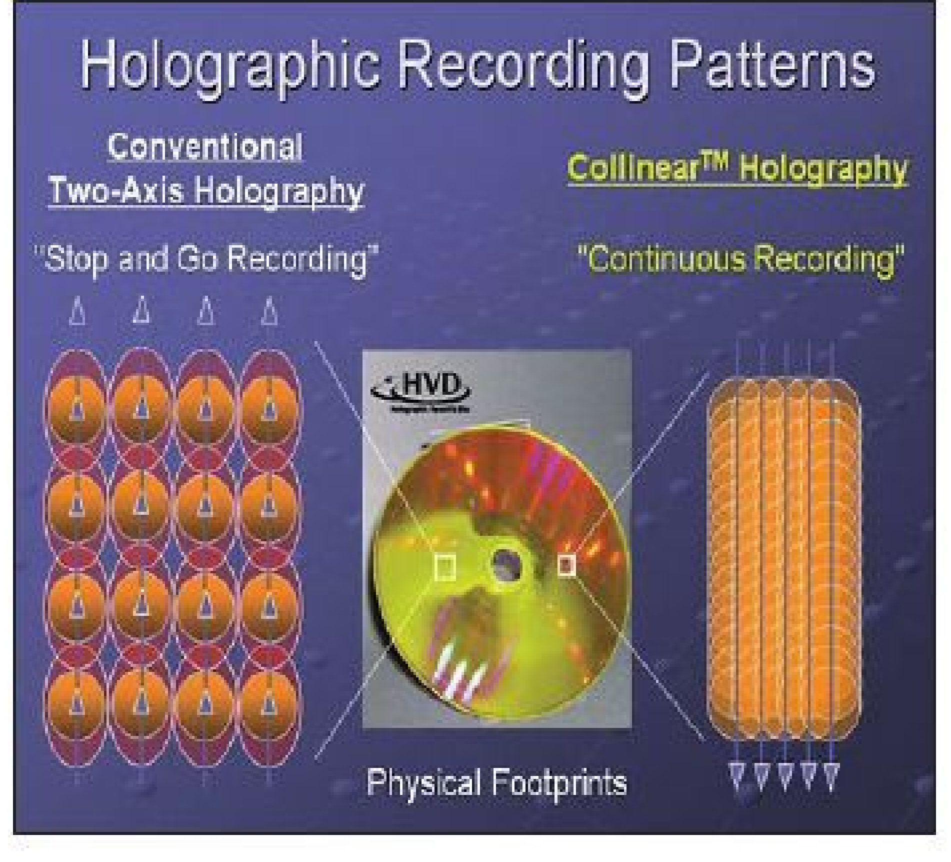
Sin lugar a dudas, los psiquiatras y psicólogos van a encontrarse con casos desorientadores cómo éste en el futuro.

[1] http://www.burbia.com/node/1703

Más allá del Blu-ray Disc: el HVD

Toda la Biblioteca del Congreso de los Estados Unidos cabría en seis de estos discos que emplean la tecnología holográfica para almacenar mayor cantidad de información. Aunque hace pocas semanas que sabemos el nombre del sucesor del DVD gracias al abandono por parte de Toshiba de su propuesta, el HD-DVD, al Blu-ray Disc de Sony ya se le está buscando un futuro sucesor. Una de estas propuestas es el HVD (Holographic Versatile Disc), con discos que pueden llegar a contener 3,9 Terabytes de información. Éste tipo de tecnología utiliza dos láseres, uno de luz verde y otro de luz roja, ambos unidos en un sólo haz. Con el primero se lee la capa holográfica de la superficie del disco, mientras que el segundo (el rojo) sirve cómo referencia y para la lectura de la capa de aluminio idéntica a la qué contienen los CD's actuales.





Con el empleo de esta tecnología un disco HVD puede llegar a contener 3,9 Terabytes de información, lo que supondría 5.800 CD-ROM's, o 850 DVD's o 160 Blu-ray Discs. La ratio de transferencia es de 1 Gigabit por segundo.

Para poner un ejemplo de la alta capacidad de almacenamiento de estos discos, se estima que la Bibilioteca del Congreso de los Estados Unidos (considerada la mayor del mundo) ocupa unos 20 Terabytes escaneada en formato texto, sin incluir las imágenes de los libros. Podría ser almacenada en seis discos... Y si quisiéramos guardar vídeo, podríamos almacenar suficiente mediante los estándares actuales cómo para un año de visualización ininterrumpida.

El HVD cuenta también con un rival, el Tapestry Media, pero por el momento éste último formato "solamente" alcanza a poder almacenar 1,6 Terabytes de datos, además de que el formato HVD se encuentra respaldado por un foro[1] de empresas entre las que se cuentan gigantes cómo Fuji, Konica o Mitsubishi, básicamente todas ellas japonesas.



Se desconoce cuando podremos adquirir los grabadores, reproductores y discos de éste sistema, aunque lo que es seguro es que aún tardaremos años si es que no se le adelanta ninguna otra propuesta y se da una "guerra" cómo la que hemos vivido entre Blu-ray Disc y HD-DVD.
Más información:
Artículo sobre HVD en la Wikipedia (en inglés) http://en.wikipedia.org/wiki/Holographic_Versatile_Disc
[1] http://hvd-forum.org/

Fundada organización independiente para garantizar la continuidad y apertura de OpenSocial



Google, Yahoo! y MySpace rubrican el acuerdo que da paso a un nuevo proyecto abierto, esta vez para las aplicaciones sociales online.
La historia de OpenSocial[1] se remonta a Noviembre del pasado 2007, cuando Google daba a conocer una API (Application Program Interface) cuya intención era y es la de facilitar el acceso y la interacción con la mayor cantidad posible de servicios sociales existentes en la Internet actual, casi todos nacidos al calor de la Web 2.0 . Desde su lanzamiento muchos servicios populares -especialmente redes sociales- se han apuntado a su carro, cómo XING, LinkedIn, Friendster o Viadeo.
La necesidad de una forma de trabajar universal a los distintos servicios viene dada por la abundancia de los mismos y las diferencias a la hora de acceder a ellos. Así, para gestionar las diferentes redes de networking a las que estamos suscritos, deberemos hacer login en todos y cada uno de los servicios utilizados. La intención de OpenSocial es ahorrar todo éste trabajo al usuario y centralizar los cambios en un sólo servicio. También se posibilita a los programadores la creación de servicios compatibles con aquellos que proporcionan sitios web muy conocidos y populares, de forma que puedan crear sus propias redes sociales pero que estas no queden en un hueco minoritario ni desconectadas del resto. Mediante el presente anuncio, Google renuncia al control directo sobre OpenSocial para basarlo a partir de ahora y en adelante en la comunidad, al estilo de otros proyectos de programación tan conocidos cómo Firefox, OpenOffice u OpenSolaris, que son construidos por una comunidad de desarrolladores independientes aunque a posteriori sean utilizados por una gran empresa para añadirles algunos elementos, un soporte técnico y distribuirlos cómo propios.
Para cumplir con éste objetivo se ha constituido la Fundación OpenSocial[2], una entidad independiente y sin ánimo de lucro que tendrá potestad sobre el desarrollo técnico, la documentación y la propiedad intelectual de OpenSocial.
Su presentación en sociedad ha tenido tres padrinos de lujo: Google, cómo compañía que ha puesto los cimientos de OpenSocial, MySpace, cómo uno de los servicios de red social más importantes de Internet y que desde el principio ha apoyado a esta iniciativa, y de forma un tanto sorprendente Yahoo!, que desde el regreso de Jerry Yang y la oferta que por ella hizo Microsoft, parece volver a la vida tras un largo letargo.
El trabajo que le queda ahora por delante a esta nueva organización es, a corto plazo, administrativo; sobre el papel ya existe, ahora solamente resta darle caras humanas para que pueda ser dirigida y empiece a trabajar.
Más información:
Nota de prensa emitida por Yahoo! http://yhoo.client.shareholder.com/press/releasedetail.cfm?ReleaseID=301421
[1] http://code.google.com/apis/opensocial/
[2] http://sites.google.com/a/opensocial. org/opensocial/Home



Google quiere utilizar los espacios entre frecuencias de televisión en los Estados Unidos para una nueva generación de dispositivos



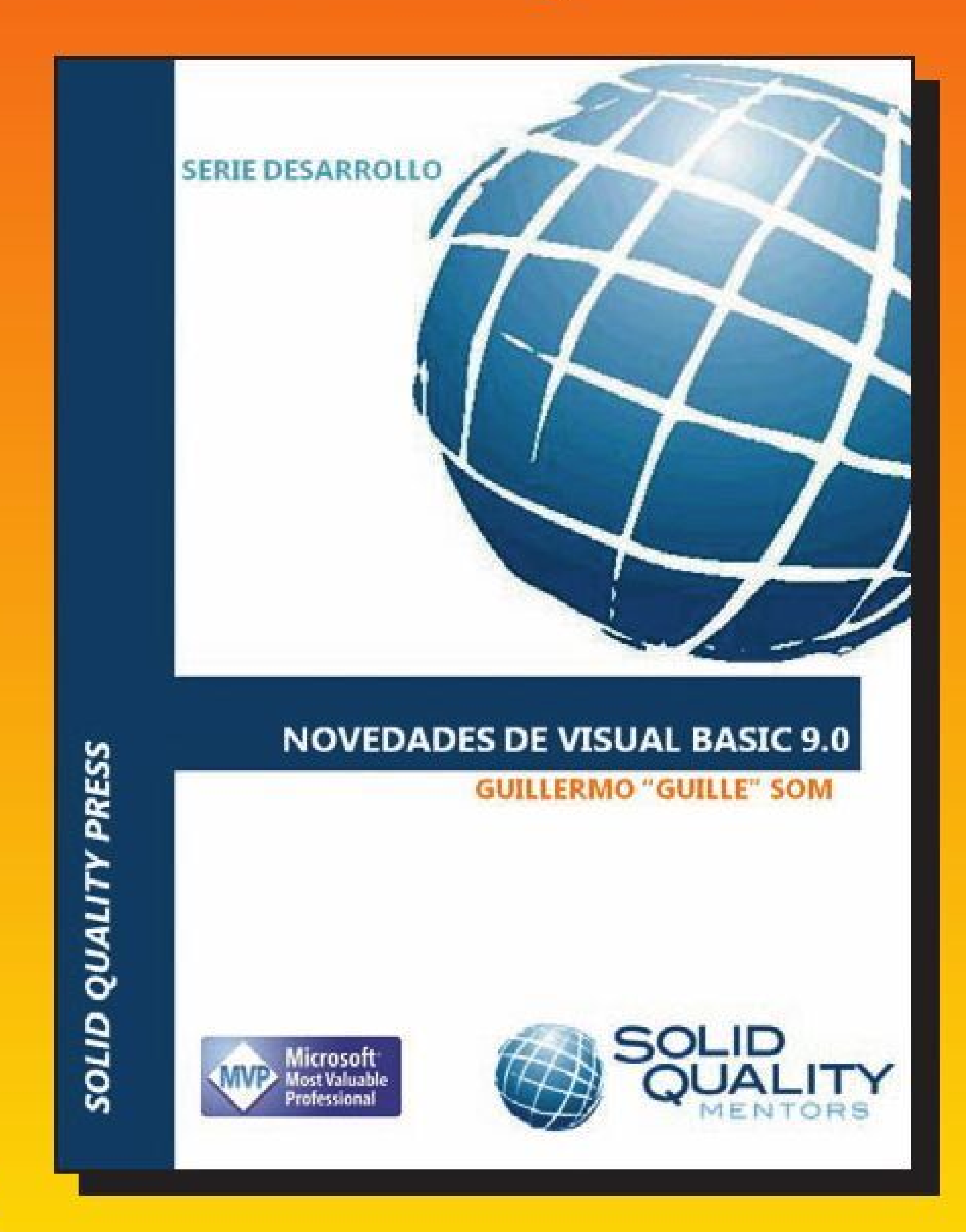
La compañía del buscador propone un conjunto formado por los espacios "en blanco" no utilizados por las emisiones de TV y su plataforma para móviles Android con el fin de proporcionar cobertura a todo el territorio estadounidense en materia de telecomunicaciones.
Pese a su derrota en las subastas para las nuevas frecuencias radioeléctricas del espacio de telecomunicaciones estadounidense, Google parece empeñada en entrar en éste sector cueste lo que cueste. Y es por ello que ha presentado una propuesta a la FCC (Federal Communication Commission, el órgano encargado de regular las telecomunicaciones en el país norteamericano) para utilizar los espacios entre canales de televisión.
Estas frecuencias, a las que Google se refiere cómo "espacios en blanco" se sitúan a modo de diferenciador entre canal y canal de la TV analógica para que los diversos canales existentes no se interfieran mutuamente. La propuesta de Google consiste simple y llanamente en utilizar dichos espacios para hacer pasar datos que serían enviados y recibidos por dispositivos equipados con Android, argumentando que al ser esta última una plataforma abierta, cualquier fabricante, prestador de servicios u operadora podría adherirse a la iniciativa.
Lo que propone Google tiene un parecido con WiMAX, un tipo de conexión inalámbrica que a diferencia del WiFi permite cubrir radios mucho más amplios, de kilómetros.
Pese a que la nota a la FCC ha sido dirigida por el gigante de Internet, éste forma parte de la White Spaces Coalition, un conjunto de grandes empresas que abogan por el uso de los ya explicados "espacios en blanco" para telecomunicaciones. Entre sus integrantes se cuentan a Dell, HP, Philips, Samsung o, paradójicamente, Microsoft.

Autor: Guillem Alsina
(guillem@imatica.org)

SORTEAMOS EL LIBRO
“NOVEDADES DE VISUAL BASIC 9.0”

Con el fin de mejorar la publicación, sorteamos este libro entre todos los lectores que nos envíen un correo electrónico con sus datos completos, indicando las fortalezas y debilidades que encuentra en la revista, sugerencias, comentarios, etc.
El ganador del sorteo se dará a conocer en el próximo nº 163 de Sólo Programadores.

Novedades de Visual Basic 9.0 contiene toda la información sobre la nueva versión del compilador de Visual Basic que se incluye en Visual Studio 2008. Novedades que se desglosan en tres partes, en la primera se cuenta todo lo referente al entorno de desarrollo, qué novedades se incorporan y cómo aprovecharlas desde el punto de vista del programador de Visual Basic. En la segunda se explican con todo lujo de detalles, todas las novedades del lenguaje, desde las nuevas instrucciones y opciones del compilador, hasta las novedades "necesarias" para trabajar con la nueva tecnología de las consultas LINQ; tema que se cubre a fondo en la tercera parte del libro, centrándose principalmente en las instrucciones de consulta incluidas en el compilador para dar soporte a las diferentes tecnologías relacionadas con LINQ: LINQ to Objects, LINQ to XML, LINQ to DataSet y LINQ to SQL.



Guillermo “Guille” Som
Es conocido en Internet por su portal dedicado exclusivamente a la programación, principalmente con Visual Basic y todo lo relacionado con punto NET, (http://www.elGuille.info/). Desde noviembre de 1997 es reconocido por Microsoft como MVP (Most Valuable Professional) de Visual Basic. Es orador internacional de Ineta con la que imparte charlas en muchos países de Latinoamérica y es mentor de Solid Quality Mentors, la empresa líder en consultoría y formación. Ha publicado dos libros con Anaya Multimedia: Manual Imprescindible de Visual Basic .NET y Visual Basic 2005.

Ficha del libro:
Título: Novedades de Visual Basic 9.0
Autor: Guillermo "Guille" Som
Número de páginas: 204
Formato: PDF, DIN A4
ISBN: 978-84-936417-0-2
Editorial: Solid Quality™ Press
Precio recomendado: 15,00 euros + 4% IVA (existen cupones de descuento)
Fecha publicación: Marzo 2008
Materia: Lenguajes de programación
Nivel: Medio / Avanzado
Tipo: Aprendizaje / Referencia





Actualidad Java de la mano de javaHispano

Sun Microsystems llevará Java al iPhone



Steve Jobs hasta la fecha ha mostrado una actitud bastante hostil ante la posibilidad de incorporar Java en uno de los productos estrella de la compañía: el iPhone. El pasado marzo, en un evento orientado a desarrolladores Apple presentó el kit de desarrollo de aplicaciones para este dispositivo, que hasta aquel momento no permitía instalar ninguna aplicación adicional a aquellas con las que venía de fábrica (excepto si el usuario estaba dispuesto a hackear el terminal, con el consecuente riesgo de perder la garantía).

Tras analizar las posibilidades del kit de desarrollo, así como todas las implicaciones legales derivadas de esta acción, Sun Microsystems ha decidido construir una máquina virtual Java para el iPhone. Estará basada en Java ME y dará acceso a toda aquella funcionalidad nativa del terminal móvil que sea posible. Además de permitir ejecutar la gran cantidad de juegos disponibles actualmente para Java ME, la máquina virtual abrirá las puertas al desarrollo de aplicaciones empresariales, punto que hasta la fecha sigue siendo la principal debilidad del producto de Apple.

Además, esta máquina virtual permitirá llevar al terminal móvil funcionalidad de la cual no dispone actualmente pero que sí está presente en Java como, por ejemplo, transmisión segura de datos a través de SSL. Sun espera tener una primera versión de su máquina virtual disponible en junio del presente año. Hasta la fecha, Apple se ha abstenido de realizar comentarios sobre este movimiento.

Sun contrata a dos líderes del proyecto Jython

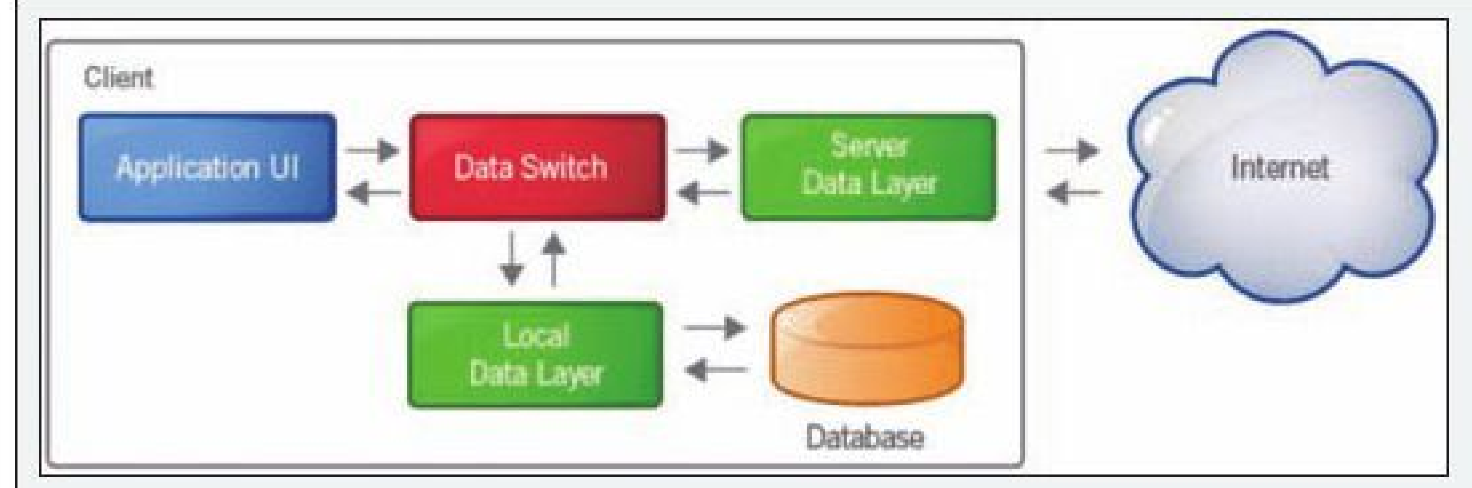


Sun Microsystems ha contratado a Ted Leung y a Frank Wierzbicki, los dos principales líderes del proyecto libre Jython, proyecto que está creando un porte de Phyton a Java de un modo similar a como JRuby creó un porte de Ruby capaz de ejecutarse en una máquina virtual Java. Al igual que sucedió con JRuby, Sun ha contratado a los dos principales desarrolladores del proyecto.

Esto permitirá a Wierzbicki y a Leung dedicar más tiempo a Jython, por lo que es de esperar que a partir de ahora lo veamos evolucionar mucho más rápido. También, probablemente, suponga que dentro de poco Netbeans proporcione un excelente soporte para Python, de un modo similar a como se convirtió en él (probablemente) mejor entorno de desarrollo para Ruby. Si a esto añadimos que recientemente ha añadido soporte de primera clase para JavaScript, Netbeans parece ir en la ruta de convertirse en el entorno de desarrollo por excelencia para lenguajes de script.



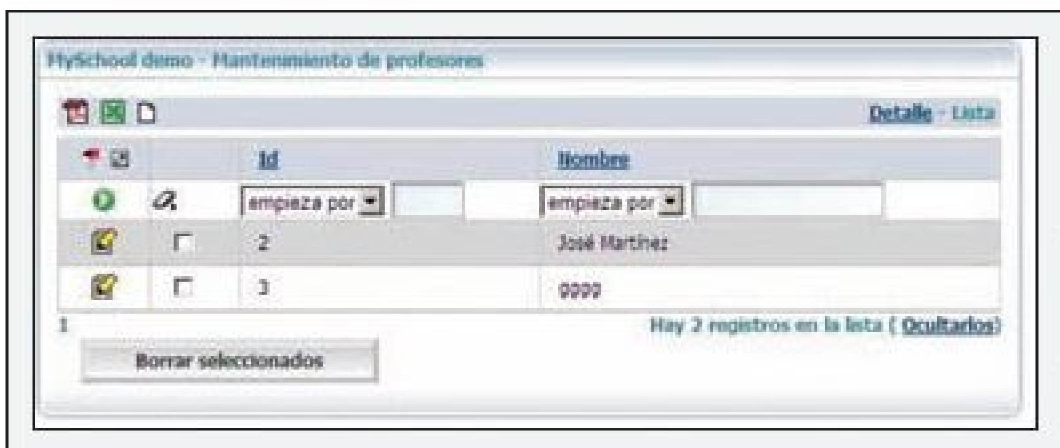
Google Gears para terminales móviles



Google Gears es una tecnología que permite la ejecución de aplicaciones web sin necesidad de conectividad. Para ello se instala como una extensión en el navegador web, extensión que incluye la popular base de datos empotrada SQL Lite. La aplicación debe usar la API JavaScript de Gears para guardar datos en esta BD y así el usuario puede trabajar con dichos datos offline.

Ahora esta tecnología se ha vuelto móvil. Google ha anunciado una versión de Google Gears orientada a navegadores de terminales móviles. Con este movimiento el gigante de Internet se adelanta a otras soluciones para construir aplicaciones RIA capaces de funcionar offline, como Adobe Air y Microsoft Silverligh, en el nicho de los clientes móviles.

Liberado OpenXava 3.0



OpenXava es un framework web que hace un gran énfasis en la productividad del desarrollador. El framework se encarga de generar la aplicación web al completo a partir de un POJO con anotaciones JPA (ver el código junto estas líneas, en el cual se han omitido los métodos getters y setters); a partir de estos objetos del modelo el framework genera automáticamente una aplicación web capaz de realizar operaciones CRUD sobre objetos del modelo; realizar búsquedas, ordenaciones y validaciones; y exportar los datos a PDF o a Excel.
OpenXava ha sido desarrollado por la empresa española Gestión Cuatrocientos y su licencia permite el desarrollo de aplicaciones comerciales, siempre y cuando no se modifique el propio código de OpenXava.

```
import javax.persistence.*;
import org.openxava.annotations.*;
@Entitypublic
class Profesor {
        @Id @Column(length=5) @Required
        private String codigo;
        @Column(length=40) @Required
        private String nombre;
}
```

SimpleJPA: JPA para Amazon Simple DB

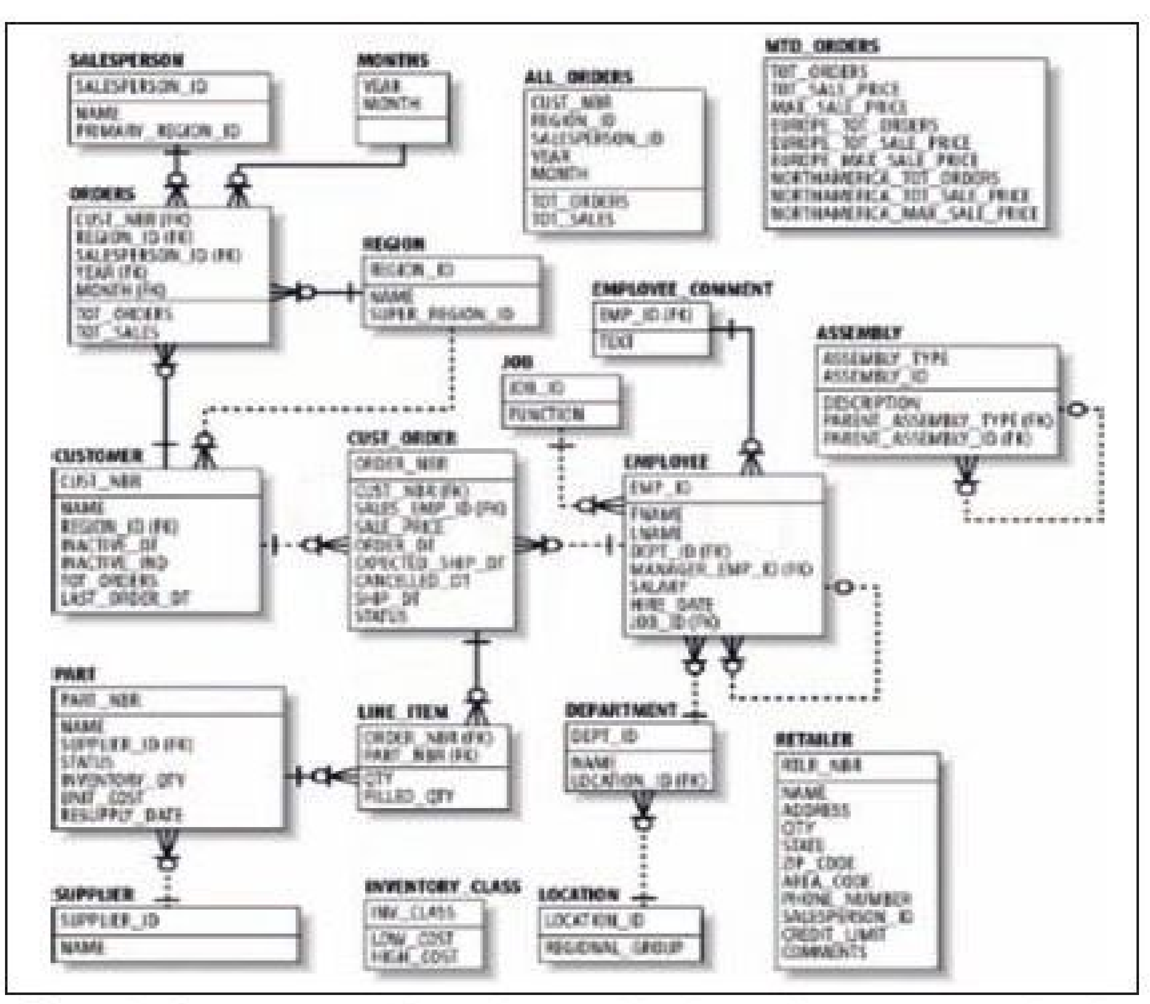


Amazon Simple DB es un servicio web de Amazon para albergar una base de datos en su "nube computacional", de forma que sólo se paga por la cantidad de uso que se haga del servicio, consiguiendo así una gran escalabilidad sin necesidad de tener un cluster de bases de datos propio.
SimpleJPA es una librería libre bajo la licencia Apache 2.0 que implementa JPA sobre Simple DB. Como su autor indica, las limitaciones de Simple BD (no es relacional, no es estructurada y no permite esquemas) hacen prácticamente imposible que algún día se implemente toda la especificación JPA, sin embargo proporciona un soporte bastante aceptable. Así, por ejemplo, soporta referencias muchos a uno y uno a muchos con lazy loading, LOBs (a través del servicio S3 de Amazón) y proporciona una caché de peticiones.

OPINIÓN

¿Sigue teniendo sentido almacenar un modelo de objetos en tablas relacionales?

La solución a la persistencia de datos ha estado tradicionalmente ligada a las BBDD relacionales. La transición a la POO, que bien podríamos dar por finalizada, no ha cambiado en absoluto este hecho.
Las diferencias entre el modelo de objetos y el relacional nos causa más de un problema a los desarrolladores. Estamos obligados a crear y mantener dos modelos (el de objetos y el relacional), así como a desarrollar clases que transformen nuestros datos de un modelo a otro.



Con el tiempo muchos desarrolladores han renegado de incrustar sentencias SQL en su código Java, pasando a emplear mapeadores O-R como Hibernate, TopLink, etc. Estos frameworks han permitido delegar el problema de la falta de concordancia objeto-relacional en una librería externa que nos hace casi transparente la traducción de un modelo a otro.
Algunos críticos de este tipo de frameworks consideran que no es el enfoque correcto para solucionar el problema. Debería considerarse la posibilidad de almacenar nuestros objetos en BBDD orientada a objetos, librándonos del problema que pretenden resolver los frameworks de mapeo O-R.
Considero que el que estemos dejando de incrustar SQL en nuestro código en favor de distantas APIs de persistencia, facilitará una futura transición a este tipo de BBDD, cuando estos sistemas maduren y nos hayamos puesto de acuerdo en un lenguaje estándar de consulta para el modelo de objetos (que podría ser algo similar a LINQ o HQL, por ejemplo).

Cristian González Losada (cristian@nevian.es)
Analista-programador en Nevian Sistemas

Sobre el autor

Abraham Otero (abraham.otero@javahispano.org) es responsable de calidad y miembro de la junta de javaHispano.



Primeros pasos con Android (y III)

AITOR ALMEIDA ESCONDRILLAS Y DAVID SAINZ GONZALEZ

Una vez vistas en los artículos anteriores unas funciones más básicas de esta plataforma Android, veamos ahora cómo aprovechar características más avanzadas.

Una aplicación más avanzada

Esta es la tercera y última entrega de la serie. En las anteriores hemos aprendido cómo realizar aplicaciones y su flujo completo. Ahora vamos a ver algunas características más avanzadas para tener una visión general del funcionamiento básico de la plataforma. En este artículo veremos cómo acceder a los mecanismos de almacenamiento del dispositivo y haremos un repaso rápido a algunas librerías opcionales.

Cómo leer, guardar y trabajar con datos

En cualquier aplicación es muy común tener que acceder a datos, guardarlos, leer y trabajar con ellos. Cada vez más se usan estas características en teléfonos móviles debido entre otros factores al aumento de las capacidades de los mismos. Android provee una forma muy sencilla de acceso a memoria secundaria, ofreciendo varias alternativas dependiendo de las necesidades:

- **Preferencias:** Así como en Java hay mecanismos especiales para guardar preferencias, en Android hay clases y métodos específicos que facilitan el guardar preferencias y opciones.
- **Archivos:** La lectura y escritura de archivos no está limitada en esta plataforma. Existen clases para leer y guardar archivos en una ruta concreta.
- **Bases de Datos:** Android ofrece la posibilidad de acceder a un pequeño motor de Bases de Datos desde el propio dispositivo.
- **ContentProvider:** esta es una funcionalidad específica de las aplicaciones que se escriben en esta plataforma, de forma que pueden guardarse datos que a la vez se compartan entre aplicaciones. Así, se exponen los datos privados de una aplicación al exterior pudiéndose colocar restricciones.

Preferencias

Para guardar preferencias tenemos clases específicas que ya realizan estas tareas. De esta forma nos ahorramos el trabajo de desarrollar mecanismos especiales en cada aplicación.

Es posible mantener preferencias privadas en el contexto de la actividad o compartirlas entre actividades del mismo paquete. Sin embargo, no será posible compartir preferencias entre diferentes paquetes.

Para recoger y editar preferencias relativas únicamente a la actividad, se utiliza el método Activity.getPreferences(). Para utilizar las preferencias compartidas dentro de un paquete, usaremos Context.getSharedPreferences(), al cual le hemos de pasar el nombre del archivo de preferencias, para que otras clases sepan a qué archivo deben acceder. Ambos métodos nos devolverán un objeto SharedPreferences que representa las preferencias. Una vez obtenidas, podemos leerlas utilizando métodos de la clase SharedPreferences como getBoolean(), getInt() etc. Para editarlas y escribir valores, utilizaremos la clase Editor, tal y como se muestra en el listado 1.

Como se observa en el listado, es necesario pasar a las funciones de acceso a preferencias un entero que simboliza el modo de acceso. Lo más normal es utilizar el cero para indicar acceso privado. También es necesario confirmar todo cambio que se haga a las preferencias a través del método commit();

Logotipo de la plataforma Android.

Material complementario

El material complementario puede descargarse desde nuestra web www.revistasprofesionales.com

LISTADO 1 — Escritura de propiedades

```
public class Actividad extends Activity
{
        public static final String NOMBRE_PREF = "OpcionesMiPrograma";

        public boolean getActivo()
        {
            SharedPreferences settings = getSharedPreferences(NOMBRE_PREF, 0);
            boolean activo = settings.getBoolean("estaActivo", false);
            return activo;
        }

        public void setActivo (boolean activo)
        {
              SharedPreferences settings = getSharedPreferences(NOMBRE_PREF, 0);
              SharedPreferences.Editor editor = settings.edit();
              editor.putBoolean("estaActivo", activo);

              //Nunca hay que olvidar confirmar los cambios
              editor.commit();
        }

        .....
}
```

Archivos

Es sencillo leer o escribir archivos. Se utilizan para esto funciones muy básicas como Context.openFileOutput() para acceso a escritura, y Context.openFileInput() para acceso a lectura. Se le pasa como parámetro el nombre y ruta del archivo y el modo de lectura (MODE_APPEND, MODE_PRIVATE etc).

Estas funciones devuelven un FileInputStream que se trata de igual forma que en Java.

Bases de Datos

De forma básica, se puede acceder a un motor de Bases de Datos dentro del dispositivo, utilizando SQLLite. Debido a las características tan básicas, se trata de una forma muy simple de Base de Datos, pero por ello mismo muy fácil de manejar.

Creamos y abrimos la Base de Datos con una simple función: Context.createDatabase() o Context.openDatabase() respectivamente. Se le ha de pasar entre otros parámetros el nombre del archivo de la propia Base de Datos, que será único dentro del paquete entero. Este archivo físicamente en el dispositivo será guardado en "/data/data/[nombre_del_paquete]/databases".

Las funciones devuelven un objeto del tipo SQLiteDatabase con el que poder trabajar.

Con esta clase, podremos hacer las operaciones básicas de una Base de Datos, como ejecutar sentencias SQL, crear, actualizar y borrar filas, recuperar datos etc.

- **execSQL():** Esta función recibe un string con una sentencia SQL que no sea una consulta y la ejecuta.
- **insert():** Tiene el mismo efecto que pasar una INSERT al método createSQL() pero tiene la ventaja de poder pasar parámetros y controlar lo que se quiere insertar. Acepta como parámetros el nombre de la tabla y una clase ContentValues, que es en realidad un mapa donde colocar los valores a insertar. Como en cualquier mapa, se colocan valores en este ContentValues con su método put(nombre_columna, valor).
- **delete():** Borra filas de una Base de Datos, aceptando tres parámetros: el nombre de la tabla, la condición where de borrado y un array con los valores que hacen que la condición se cumpla.
- **update():** Actualiza datos en una fila, utilizando cuatro parámetros: el nombre de la tabla, una clase ContentValues para colocar los pares columna-valor, y otros dos parámetros parecidos a los de la función delete: condición where y array de valores que hacen que la condición where se cumpla.
- **query():** Hace una selección de filas de la tabla de forma que colocamos una sentencia SELECT repatida en parámetros. Tiene varias variantes, una de ellas acepta estos parámetros:
  - Booleano para definir si queremos una SELECT distinct.
  - String para dar el nombre de la tabla.
  - Array de Springs para definir el nombre de las columnas que queremos retornar.
  - String conteniendo la cláusula WHERE (sin declarar explícitamente la palabra WHERE).
  - Array de Springs para incluir una lista de argumentos, en caso de que hayamos incluido argumentos (con el carácter '?') en la cláusula WHERE.
  - String con la cláusula GROUP BY (sin declarar explícitamente las palabras GROUP BY)
  - String con la cláusula HAVING (sin declarar explícitamente la palabra HAVING)
  - String con la cláusula ORDER BY (sin declarar explícitamente las palabras ORDER BY)
- **rawQuery():** ejecuta una consulta a la Base de Datos pasando directamente un string con la sentencia SQL y un array de Springs con los parámetros de la sentencia en caso de haberlos utilizado.

Estos últimos métodos devuelven un conjunto de filas como resultado de la consulta. La forma de devolverlas es similar a la utilizada en Java. Se devuelve un objeto Cursor que contiene sus métodos de close(), count(), first(), next(), getInteger(), getString() etc.

El listado 2 muestra un ejemplo de uso de acceso a Bases de Datos con Android.

Content Providers

Todos estos métodos que acabamos de ver son buenas y efectivas formas de guardar datos en el dispositivo, sin embargo su uso es eminentemente privado para la propia aplicación o paquete. Si queremos hacer nuestros datos públicos a otras aplicaciones, hemos de utilizar otro mecanismo, llamado Content Provider. Esta es la forma que tiene esta plataforma de compartir datos entre aplicaciones.

En realidad, un Content Provider no es otra cosa que una interfaz, un mecanismo para decir cómo se ha de acceder al contenido. Realmente la implementación del almacena-

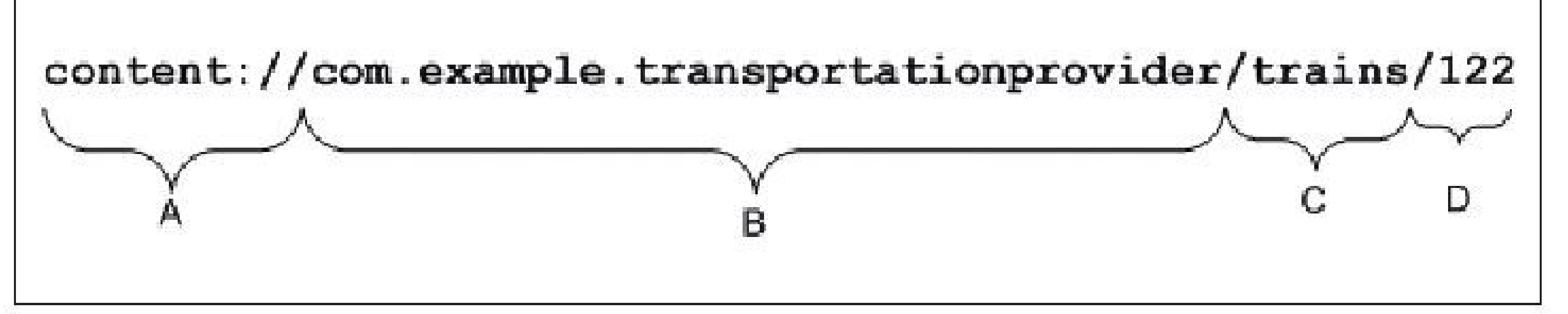


Figura 1.



miento de datos ha de hacerla quien crea el Content Provider específico. Normalmente aquél que hace un Content Provider a su vez se basa en los mecanismos vistos para almacenar y recuperar realmente los datos.

Android tiene por defecto unos Content Providers por defecto para acceder a ciertos cometidos como audio, video, imágenes, contactos personales, etc. Que pueden usarse por cualquier aplicación.

Usar Content Providers

La forma de obtener datos en Android es siempre a base de URIS, con ellas identificamos lo que queremos obtener. Para acceder al contenido de un Content Provider, se utiliza una URI que siempre comienza por "content://". Por ejemplo, para acceder a una persona especifica del equipo azul, se puede utilizar "content://com.solop.personas/equipoazul/15" y para acceder a una lista que muestre el equipo azul entero se utiliza "content://com.solop.personas/equipos/equipoazul/". Es interesante saber que esto nos aísla una vez más de la aplicación que realmente está ejecutando el acceso a estas personas, sólo introducimos una URI y obtenemos el objeto que deseamos. (Ver Fig. 1).

Formato de una URI para contenido. A: cabecera. B: Authority. C: Parte para identificar de qué tipo de datos se trata. D: Un registro específico.

Al ver la URI para obtener a un participante concreto del equipo azul, vemos que estamos solicitando el identificador 15. Esto es porque cualquier elemento que recogemos tiene por fuerza un campo de identificador. En el momento de hacer nuestro propio Content Provider (que veremos un poco más adelante) tenemos que tener esto muy en cuenta.

Los pasos para la consulta de datos son básicamente crear la URI del contenido que queremos consultar y pasarla al Content Provider para recoger el objeto.

Casi siempre la propia clase del Content Provider nos va a proporcionar la URI necesaria en una variable llamada CONTENT_URI, con lo que no será necesario crearla a mano. Si acaso podemos añadirle el identificador buscado si lo que estamos buscando es un elemento concreto:

```
Uri persona = solop.provider.personas.
equipos.azul.CONTENT_URI;
persona.addId(2);
```

LISTADO 2 — Acceso a Bases de Datos

```
public DatabaseManager(Context ctx)
{
        try
        {
             db = ctx.openDatabase(NOMBRE_BD, null);
        }
        catch (FileNotFoundException e)
        {
             try
             {
                  db = ctx.createDatabase(NOMBRE_BD, DATABASE_VERSION, 0, null);
                  db.execSQL("CREATE TABLE PERSONASEQAZUL (_id INTEGER PRIMARY KEY
 AUTOINCREMENT, NOMBRE TEXT NOT NULL, APELLIDOS TEXT);");
             }
      catch (FileNotFoundException e1)
      {
                  db = null;
             }
        }
}

public List<Persona> getPersonas()
{
        ArrayList<Persona> ret = new ArrayList<Persona>();
        try
        {
             Cursor c = db.query(TABLA, new String[] {"_id", "nombre", "apellidos"},
                            null, null, null, null, null);
             int numRows = c.count();
             c.first();
             for (int i = 0; i < numRows; ++i)
             {
                  Persona persona = new Persona();
                  persona._id = c.getLong(0);
                  persona.nombre = c.getString(1);
                  persona.apellidos = c.getString(2);
                  ret.add(persona);
                  c.next();
             }
        }
        catch (SQLException e)
        {
             Log.e("Excepción en Database Manager", e.toString());
        }
return ret;
}

public int updatePersona(long id, String nombre, String apellidos)
{
        ContentValues args = new ContentValues();
        args.put("nombre", nombre);
        args.put("apellidos", apellidos);
        return db.update(TABLA, args, "_id=" + id, null);
}
```

Con estas líneas de código ya hemos conseguido una URI como esta: "content://com.solop.provider.personas/equipos/azul/2".

El siguiente paso es recoger el recurso. Para ello disponemos básicamente de 2 formas:

- **Recogerlo en forma de Cursor.** Llamando a una función de la clase Activity llamada Activity.managedQuery(). Esta función recibe la URI que identifica al recurso (además de las columnas que queremos recoger) y devuelve un objeto de la clase Cursor con la que poder trabajar. En el Listado 2 ya tenemos un ejemplo de uso de cursor.
- **Recogerlo en forma binaria.** Si se guardan archivos binarios, pueden recogerse a través de la función ContentResolver.openInputStream(). Acepta la URI del recurso y devuelve un InputStream sobre el que leer. También hay una versión para OutputStream.

Para modificar datos, podemos hacerlo a través de modificación del cursor. Además, si utilizamos el acceso a los datos en forma de Cursos, al estilo Bases de Datos, necesitamos acceder a las columnas. Para ello, cada clase que represente un Content Provider, tendrá en forma de constantes el nombre de las columnas que pueden recogerse.

Crear Content Providers

Una vez entendido un poco más el concepto de Content Provider, su relación con las URIs

LISTADO 3 — Parte del código de un Content Provider

```
public class PersonaProvider extends ContentProvider
{
        //COLUMNAS
        public static String NOMBRE = "Nombre";
        public static String APELLIDOS = "Apellidos";
        public static String ID = "_id";

        //URI
        public static final ContentURI CONTENT_URI = ContentURI.create("content://com.solop.provider.personas");

        //Método interno: Base de Datos
        DatabaseManager manager = null;

        @Override
        public int delete(ContentURI arg0, String arg1, String[] arg2)
        {
                long id = arg0.getPathLeafId();
                manager = new DatabaseManager(this.getContext());
                manager.borrar(id);
                manager.close();
                return 0;
        }

        @Override
        public String getType(ContentURI arg0)
        {
                //Tipo cursor
                return "vnd.solop.cursor.item/miembroequipo";
        }

        @Override
        public ContentURI insert(ContentURI arg0, ContentValues arg1)
        {
                manager = new DatabaseManager(this.getContext());
                long id = manager.insertar(arg1.getAsString(NOMBRE), arg1.getAsString(APELLIDOS));
                manager.close();
                arg0.addId(id);
                return arg0;
        }

        @Override
        public Cursor query(ContentURI uri, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy)
        {
                manager = new DatabaseManager(this.getContext());
                Cursor c = manager.GetCursorPersonas(columns, selection, selectionArgs, groupBy, having, orderBy);
                manager.close();
                return c;
        }

.....

}
<?php
   require_once("phpFlickr/phpFlickr.php");
   // Create new phpFlickr object
   $f = new phpFlickr("Tu Flickr API Key");
   $photos_interestingness= $f->interestingness_getList();
   $i = 0;
   foreach ((array)$photos_interestingness['photo'] as $photo) {
      echo "<a href=\"photoBig.php?photo=". $photo['id'] ."\">";
      echo "<img class=\"smallPhoto\" border='0'
            alt='$photo[title]' "."src=" . $f->buildPhotoURL($photo, "Square") . " \">";
      echo "</a>";
      $i++;
      // If it reaches the sixth photo, insert a line break
      if ($i % 3 == 0) {
         echo "<br>\n";
      }
   }
?>
```

y los identificadores, podemos crear nuestros propios contenidos accesibles por otras aplicaciones, en caso de que lo necesitemos. El primer paso será primero definir qué datos queremos que sean visibles a otras aplicaciones. La aproximación más común es devolver una serie de columnas, como si se tratase de una Base de Datos. Realmente no tenemos por qué implementar por dentro nuestro Content Provider con una Base de Datos (podemos utilizar registros en un fichero, propiedades, etc.) pero la visión por fuera será a base de columnas.

Después, hemos de crear una clase que herede de ContentProvider. En esta clase



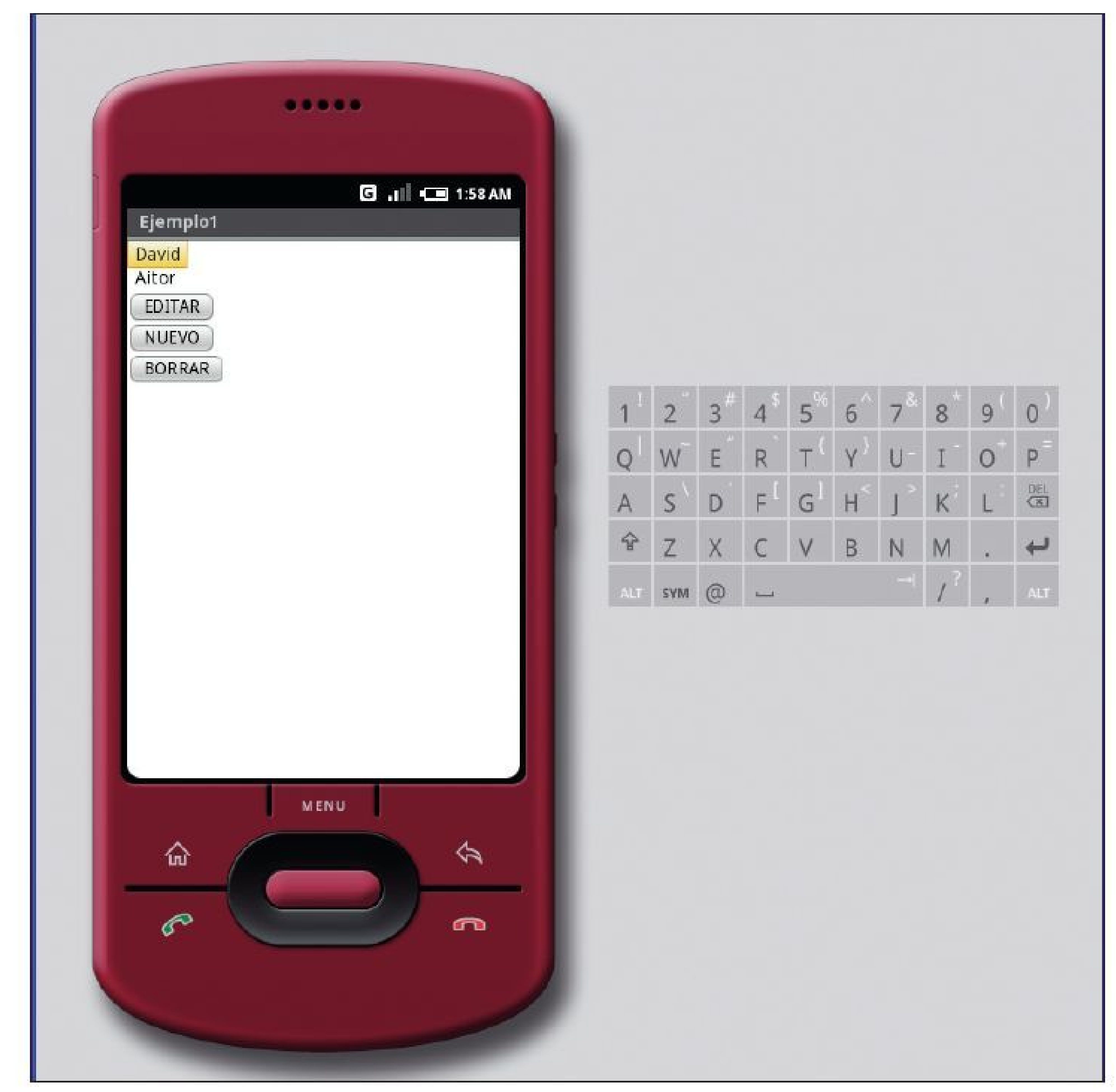
incluimos unas constantes string para referirnos al nombre de columnas que vamos a utilizar, de entre ellas una deberá ser el identificador. Así como nosotros usamos constantes para referirnos a nombres de columnas cuando utilizamos un Content Provider, ahora somos los que necesitamos crear dichas constantes. Por otra parte, hemos de pensar también en la URI que queremos para referirnos a nuestros datos. Utilizaremos una constante de esta manera:

```
public static final ContentURI CONTENT_
URI = ContentURI.create("content://
com.solop.provider.personas");
```

En la que podemos ver sólo se explicita la base de la URI.

Además de crear URI y nombres de columnas, en esta clase tiene unos métodos abstractos que debemos implementar. Básicamente dictan cómo debemos acceder al contenido desde fuera. Estos métodos son muy similares a los ya vistos en manejo de Bases de Datos y son:

- **query():** Esta función se usa para recoger los datos desde otras aplicaciones. Acepta parámetros de URI a la que nos referimos, criterios de selección, orden o columnas que queremos obtener. Siguiendo esos parámetros, debemos construir una función que entregue los datos que la aplicación externa concretamente pide.
- **insert():** función en la que hemos de implementar el código para añadir datos. Recibiremos la URI concreta del recurso y un mapa con conjuntos clave-valor, que serán el nombre de la columna y el valor que se le quiere aplicar.
- **update():** implementación de actualización de datos. Además de URI y mapa de claves-valor, recibiremos como parámetro otro String que indique qué criterio utilizar a las columnas que se actualizan (similar a una cláusula WHERE en una sentencia SQL UPDATE).
- **delete():** implementación de borrado de datos. Asimismo recibiremos la URI para ver qué recurso borrar y un String con el criterio de borrado.
- **getType():** debemos devolver el tipo MIME del contenido que manejemos. Acepta el parámetro de URI siempre para ver a qué contenido nos referimos. El formato básico de string a devolver al recibir, por ejemplo, la URI "content://com.solop.provider.personas/equipos/azul/2" podría ser "vnd.solop.cursor.item/miembroequipo".
- **onCreate():** Si se ha de hacer alguna incialización previa, puede hacerse en este método, que se llama cuando el Content Provider se está creando.



Listado de personas en la aplicación.

Teniendo en cuenta que sobre todo en el método query se trabaja con cursores, esta vez seremos nosotros los que lo crearemos. Esto no es mayor problema si nos basamos por dentro en una Base de Datos: recogemos el cursor que hayamos obtenido de SQLite y lo devolvemos. Si utilizamos otros mecanismos de almacenamiento, hemos de crear un cursor y poblarlo con datos.

Una vez creada la clase, ya tenemos un Content Provider para que otras aplicaciones accedan a nuestros datos. Ahora el último paso será incluir en el manifiesto de la aplicación una etiqueta llamada *provider* que informe que existe un Content Provider nuevo y lo queremos publicar al resto de aplicaciones.

```
<provider class="PersonaProvider"
android:authorities="com.solop.
provider.personas" />
```

En el listado 3 tenemos un ejemplo de Content Provider.

Una vez hemos realizado todas estas acciones, tenemos disponibles nuestros datos a aplicaciones externas.

Librerías adicionales

Android tiene una buena gama de librerías básicas para realizar todas las acciones necesarias en cualquier aplicación, pero además de ello, ya desde el principio nos ofrece una serie de librerías extra con las que podemos sacar mucho más partido a los dispositivos y sus características avanzadas.

Las primeras librerías son para proveer de servicios basados en localización. Entre otros, se trata de posicionamiento y mapas. El paquete en el que se encuentran las primeras es android.location. Estas clases proveen de las acciones necesarias para averiguar la localización del dispositivo en todo momento. La clase central es



LocationManager, que servirá de punto de entrada a todos los servicios basados en localización. De ella dependen los LocationProviders, que nos ofrecen la posición. Cada Location Provider utilizará un mecanismo concreto para averiguar la localización, como puede ser satélite por GPS, utilizar datos del proveedor de telefonía, utilizar posicionamiento por Bluetooth o incluso crear nuestro propio Locationn Provider.

Utilizando la clase LocationManager podremos:

- Averiguar la lista de Location Poviders que actualmente hay disponibles
- Subscribirse/Desubscribirse a las notificaciones que cada uno de ellos lanza para informar de la posición actual.
- Inscribir/Retirar un Intent concreto que pueda dispararse cuando la localización actual entre en un área previamente definida.

Esta librería ofrece muchas posibilidades a la hora de realizar aplicaciones basadas en localización, ahora tan de moda. Además, para que pueda ejecutarse en emuladores, por defecto se incluye un Location Provider de GPS que simula un recorrido entre dos puntos de San Francisco, para que puedan realizarse aplicaciones sin necesidad de GPS real.

Además de la librería de localización, la propia pltaforma ya tiene incluido un paquete llamado com.google.android.maps para trabajar con los populares Google Maps. Este paquete viene con las clases necesarias para mostrar por pantalla y manejar los mapas, además de poder controlar sus capas. Existe un View concreto llamado MapView para mostrar dichos mapas en las interfaces gráficas. La actividad que quiera colocar uno debe heredar de MapActivity en lugar de hacerlo de Activity.

Otros paquetes opcionales ofrecen también manejo de recursos multimedia. El paquete android.media ofrece clases para reproducir y grabar video y sonido. Podemos usar la clase MediaPlayer para reproducir y la clase MediaRecorder para grabar. A continuación se muestra un sencillo código para ejecutar un archivo multimedia:

```
MediaPlayer mp = new MediaPlayer();
mp.setDataSource(PATH_AL_ARCHIVO);
mp.prepare();
mp.start();
```



Edición de una persona en la aplicación ejemplo.

También nos ofrecen en Android la posibilidad de incluir gráficos 3D en los dispositivos. Teniendo en cuenta que ya muchos móviles tienen suficiente potencia para utilizar aceleración 3D, en Android es posible utilizar librerías para el uso de OpenGL, un mecanismo de aceleración 3D que puede renderizar gráficos en 3 dimensiones, de forma muy parecida a la que existe ya para los móviles Java.

Para terminar, estas librerías opcionales en un futuro cercano ofrecerán librerías para el acceso a las capacidades Bluetooth o WiFi. Actualmente no hay acceso desde librerías de alto nivel (aunque es posible hacerlo accediendo a Hardware de bajo nivel, cosa que Android nos permite hacer), así que tendremos que esperar aún un poco más para disfrutar de librerías de este tipo.

Conclusión

Android es una plataforma incipiente, pero ya en un estado de desarrollo muy avanzado y con un futuro muy prometedor. Apoyada por una compañía como Google, promete dar mucho de qué hablar en los próximos meses.

Desde esta serie de 3 artículos, hemos visto cómo crear una aplicación sencilla desde el principio, y la hemos ido complicando un poco más hasta llegar a temas algo más avanzados. Para poder tener una visión más general, en este número se incluye un pequeño ejemplo realizado en Eclipse que sintetiza todo lo visto hasta ahora. Es muy recomendable revisar su código para completar los ejemplos vistos en los listados. Una vez entendido el ejemplo, ya seremos unos iniciados en la tecnología. SP



Internet Explorer 8 calienta motores

NICOLÁS VELÁSQUEZ ESPINEL

Recientemente Microsoft ha lanzado una versión de prueba del que será su nuevo navegador Internet Explorer 8, una beta que sale cuando algunos todavía ni siquiera se han pasado a la versión 7 de este popular programa.

El anuncio se produjo durante la conferencia de tecnología online MIX08 de Microsoft. Allí, Dean Hachamovitch, gerente general de equipos de la compañía y encargado de Internet Explorer, ofreció una primera descripción del sucesor de Internet Explorer 7 (lanzado allá por octubre de 2006), que dispone por el momento de versiones para Windows Vista y Windows Server 2008, XP SP2 Profesional y Home (32 y 64 Bits) y Windows Server 2003 SP2 (32 y 64 Bits).

Destaca por incorporar varias novedades. Así, gracias a los Activities, tendremos la posibilidad de obtener información relativa a sitios webs, o incluso lugares, con tan sólo pulsar sobre una palabra y pedir a Live Maps de Microsoft -competencia directa de Google Maps- que nos muestre en un mapa información relativa a la palabra o frase que hemos señalado. Del mismo modo, podremos enviar la página que estemos visitando a nuestro blog en Windows Live Spaces con sólo un clic del botón derecho del ratón.

También dispondremos de una ayuda a la navegación con "Discover with Me.dium", un servicio de recomendación de páginas similares a la que estamos visitando basadas en la votación de los usuarios, y permite guardar en un ordenador local los trabajos que están haciendo en un sitio de Internet cuando por ejemplo se caiga la conexión a la Red. Por otro lado también da la opción a los usuarios de marcar una dirección de un sitio de web y luego ver un mapa del sitio con solo presionar un botón.

Ya puedes descargarte gratuitamente la beta de IE8 para probarla.

En lo que respecta a aplicaciones de terceros, Internet Explorer 8 incorpora los WebSlices, pequeñas pestañas que posibilitarán recibir información actualizada y personalizada de sitios web como Facebook o eBay, habilitada en forma de pequeñas ventanas con información relevante, en tiempo real.

Otras novedades incluyen mejoras en la herramienta anti-phishing, con el nombre del dominio que realmente estamos visitando destacado en negrita, tres modelos de "rendering" de páginas diferente para lograr una mayor compatibilidad, y más herramientas de desarrollo a disposición de los programadores para depurar HTML, CSS y los scripts en un entorno visual de una manera rápida.

Como prioridad se quiere implementar todos los soportes de RSS, CSS 2.1 y AJAX dentro del proyecto, así como incorporar sus propias aplicaciones en los menús contextuales del navegador, consciente del dominio de su principal rival, Google, contra el que hay toda una estrategia planificada, como enemigo declarado.

Y es que la guerra de navegadores es un conflicto silencioso que ya ha sufrido importantes bajas. Y si no que se lo pregunten a los responsables de Netscape que recientemente anunciaban que no seguirían actualizando el producto ni darían más soporte para el mismo, aunque lo dejaban a disposición de quién quisiera seguir bajándoselo de la red. El hecho tiene especial relevancia si se tiene en cuenta que Netscape fue el primer navegador web que apostó fuerte por Internet como algo más que un fenómeno puntual, posicionándose como líder del mercado y haciéndose con una cuota de usuarios realmente importante, hasta que la empresa de Bill Gates decidió lanzar su Internet Explorer, reconociendo así implícitamente su equivocación al no apostar inicialmente por la Red (la despreció como una "moda pasajera"). Desde ese entonces y, pese a las múltiples críticas que ha recibido la herramienta de Microsoft desde entonces, Netscape ha sufrido lo indecible para resistir en un lugar honroso pero indefectiblemente condenado ya desde hace algunos años,



Microsoft confía en que IE8 suponga un salto cualitativo.

cuando alternativas como Opera o el popular Firefox de Mozilla adquirieron un protagonismo más que merecido, que por otro lado suponen un más que digno relevo generacional.

La batalla ahora se libra en un frente dual, protagonizado por Microsoft y su rival Google que respalda claramente a Firefox (aporta cerca del 85% de los ingresos de la Fundación Mozilla con 43 millones de euros en 2006). Y es que el incuestionable éxito de Google supone un claro espaldarazo a Firefox que es distribuido como su software preferido, por lo menos hasta que venza el contrato existente entre ellos, el próximo 20 de noviembre de 2008. Quizás para entonces las cosas cambien.

Por lo pronto, esta nueva versión de Internet Explorer se presenta en sociedad como una respuesta hacia el futuro Firefox 3. Hay una apuesta clara por la compatibilidad hacia atrás con las páginas no estándar desarrolladas para funcionar con Explorer 5 o 6, aunque nadie dijo que esto fuera sencillo. De momento la beta disponible básicamente te permitirá asegurarte de que un sitio web o aplicación web no tendrá problemas con sus estándares. Además, IE8 emula Internet Explorer 7 de forma que al final tendrás a mano dos exploradores en uno sin mayores complicaciones.

Seguramente habrá que esperar cerca de un año y medio para que la versión final vea la luz y que podamos comprobar si las funcionalidades aportadas están reñidas con el rendimiento o el consumo de memoria. Mientras tanto esta beta supone un excelente aperitivo para los más "techies".

Si quieres probarlo descárgalo gratuitamente desde el siguiente enlace: http://www.microsoft.com/ie/ie8 SP



RSS con Java (I)

ADOLFO ALADRO GARCÍA

La tecnología RSS ofrece otra forma de navegar por Internet que cada vez es más común. Los principales navegadores detectan automáticamente los RSS de los portales y servicios, permitiendo al usuario suscribirse. Por lo tanto la generación de RSS se ha convertido en una tarea común en el desarrollado de aplicaciones Web. En esta serie se estudiará cómo hacerlo con Java, utilizando su API estándar, de una forma robusta y eficiente.

¿Qué es un RSS?

Un *RSS* es simplemente un documento el formato *XML* que proporciona una serie de elementos de información entorno a un tema que se denomina canal. Un gran número de portales, servicios, periódicos *online*, buscadores como *Google*, servidores de noticias, etc., proporcionan ya *RSS*. Por ejemplo, en *www.technorati.com* se puede buscar *madonna* y en la página de resultados se muestra un icono con el texto *subscribe*. Al hacer clic en ese icono se accede a un *RSS*. Es un documento *XML* pero lo más normal es que el navegador lo detecte como canal *RSS* y entonces proponga al usuario realizar algún tipo de acción de acuerdo con las herramientas que dicho navegador ofrece para la lectura de la fuente *RSS*. Sin embargo si se observa el código fuente se comprueba efectivamente el formato *XML* del canal.

Existen distintos formatos *RSS*. Difieren entre ellos en varios aspectos de la estructura del documento *XML* pero en esencia representan la misma información. Uno de los formatos más simples es *RSS 2.0* y en esta serie de artículos es el que se va a utilizar. Una vez que se domina cómo generar y servir *RSS* de forma eficiente desde una aplicación *Web*, extender los desarrollos para cubrir más formatos es una cuestión menor.

El ejercicio más simple con el que se puede empezar consiste en coger un documento *XML* correspondiente a un *RSS*, como el obtenido en el ejemplo anterior, y ver cómo se puede servir desde un *servlet*. El siguiente ejercicio consistirá en estudiar cómo es posible generar dinámicamente el documento *XML* desde el servidor utilizando el *API* estándar de *Java*.

Servir RSS: primera aproximación simple

El *servlet* llamado *SRssServer1* ilustra un ejemplo muy sencillo acerca cómo servir un *RSS* que ya está hecho, y que se ha colocado en el paquete *com.spp.rss* con el nombre *madonna.xml*. Como siempre, el método principal del *servlet* se denomina *service*.

```
public void service(HttpServletRequest
req, HttpServletResponse res)
throws ServletException, IOException
{
  ...
}
```

Cuando se hacen desarrollos que trabajan con canales de entrada y salida, como es el caso, conviene emplear el típico "esqueleto" de programación que garantiza la liberación de los recursos utilizados con independencia de lo que ocurra en el bloque principal del código. Así las variables correspondientes al canal de lectura, de tipo *BufferedReader*, y al canal de escritura, de tipo *BufferedWriter*, se declaran justo antes de que comience el bloque *try...catch*, y se inicializan a *null*:

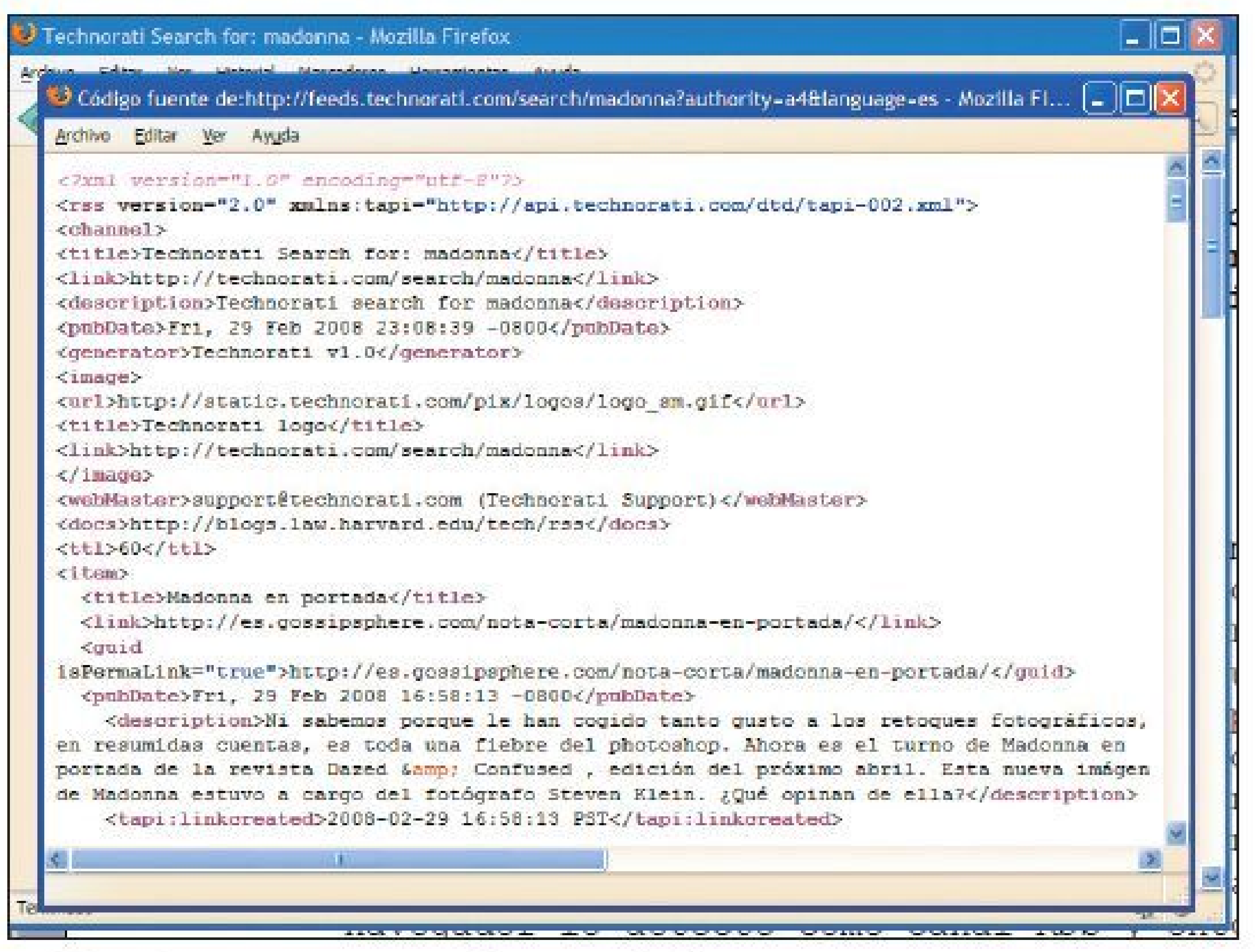
```
BufferedWriter bw = null;
BufferedReader br = null;
try {
  ...
} catch (Exception e){
  ...
} finally {
  try {if (bw!=null) bw.close(); bw=null;}
  catch (Exception e) {}
  try {if (br!=null) br.close(); br=null;}
  catch (Exception e) {}
}
```

En la cláusula *finally*, cuando el código principal ha terminado, tanto si se han producido errores como si no, los canales se cierran empleando el método *close*.

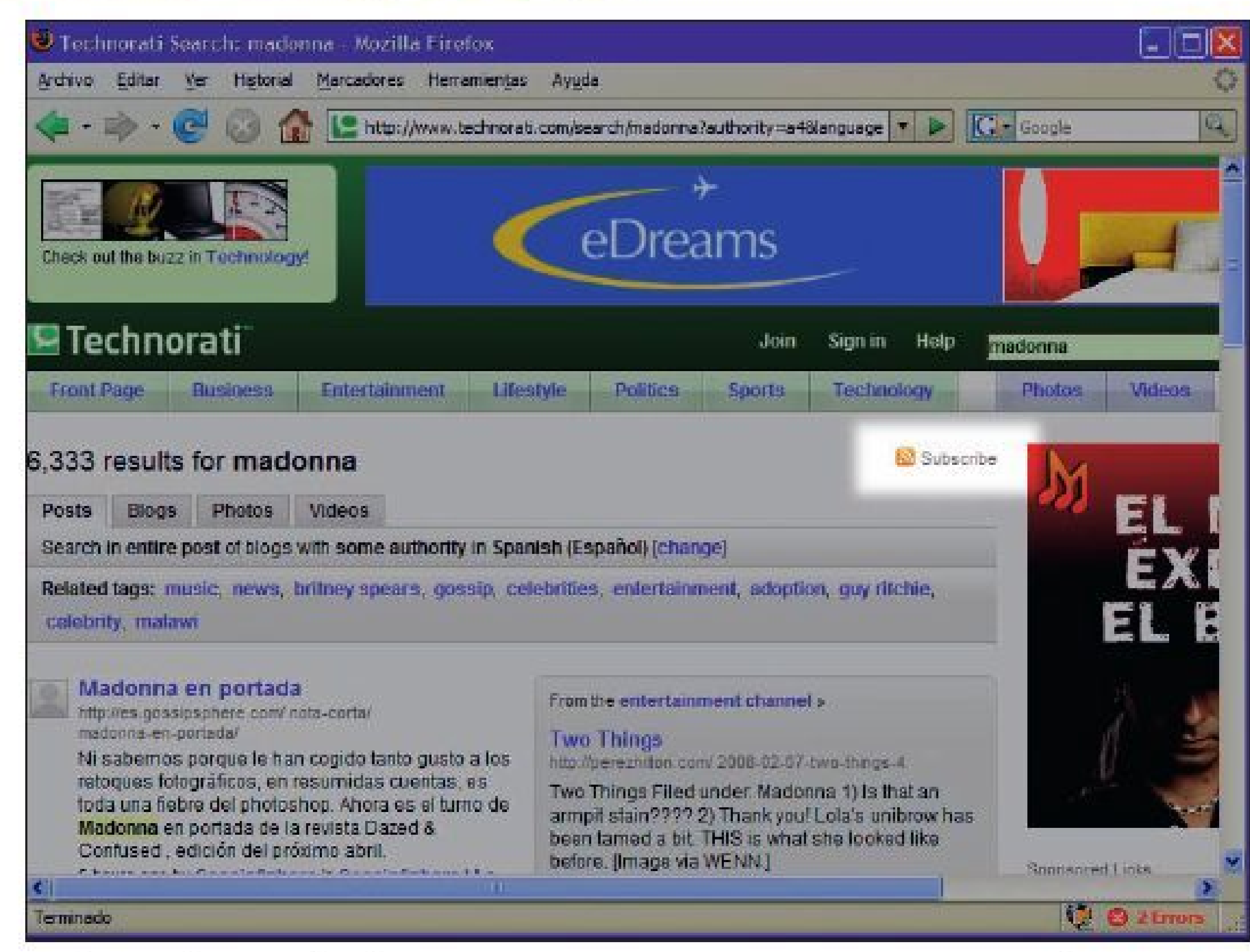
Material adicional

El material complementario puede ser descargado desde nuestra web www.revistasprofesionales.com





Código fuente correspondiente a la fuente RSS.



Página de resultados que muestra un icono para suscribirse a una fuente RSS.

El canal de lectura se emplea para leer el fichero *XML* correspondiente al *RSS*. Como el fichero se encuentra en el *classpath* de la aplicación *Web* se puede acceder a el con el método *getResourceAsStream* del objeto *Class* de la propia clase del *servlet*:

```
br = new BufferedReader(new InputStream
Reader(SRssServer1.class.getResource
AsStream("/com/spp/rss/madonna.xml"),
Charset.forName("UTF-8")));
```

El canal de escritura permite devolver el contenido al navegador, es decir, el código *XML* correspondiente al *RSS*:

```
bw = new BufferedWriter(new OutputStream
Writer(res.getOutputStream(), Charset.
forName("UTF-8")));
```

Obsérvese que el método *getOutputStream* de la interfaz *HttpServletResponse* devuelve un canal de escritura simple orientado a *bytes*, de tipo *OutputStream*. La clase *OutputStreamWriter* se emplea para transformar dicho canal en otro orientado a caracteres y adicionalmente se usa el segundo parámetro del constructor para indicar el *charset*, que lógicamente coincide con el del canal de lectura.

Antes de comenzar la lectura y escritura es necesario fijar las cabeceras *HTTP* de la respuesta:

```
res.setContentType("text/xml; charset=
UTF-8");
res.setDateHeader("Expires", -1);
res.setHeader("Cache-Control", "no-cache");
res.setHeader("Pragma", "no-cache");
```

En la primera línea se indica el tipo de datos devuelto junto con el *charset* empleado. El tipo *text/xml* es genérico y se utiliza para los documentos *XML* en general. Existen otros tipos más específicos para servir *RSS* como *application/rss+xml*, *application/rdf+xml* y *application/atom+xml* pero su funcionamiento no es robusto en el sentido de que dependiendo del lector no siempre se reconocen dichas cabeceras. El tipo *text/xml* con ser menos preciso es más fiable y compatible a día de hoy. El resto de las cabeceras se emplean simplemente para evitar la *cache* en el cliente.

Finalmente la lectura y escritura se lleva cabo utilizando un pequeño *buffer* de caracteres:

```
char[] buffer = new char[128];
int iReadChars;
...
while((iReadChars=br.read(buffer, 0,
128))!=-1) {
  bw.write(buffer, 0, iReadChars);
}
bw.flush();
```

Es muy importante hacer hincapié en el hecho de que las cabeceras *HTTP* deben establecerse antes de haber obtenido el canal de escritura para comenzar a escribir. No se trata de un requerimiento propio de la tecnología *RSS* sino de los *servlets* en general.

En el fichero *web.xml* del servidor de aplicaciones se ha establecido la siguiente configuración:

```
<servlet servlet-name="SRssServer1"
servlet-class="com.spp.rss.SRssServer1"/>
<servlet-mapping url-pattern="rss1
.xml" servlet-name="SRssServer1"/>
```

Así, para la versión local de pruebas, la *URL* *http://www.revistasprofesionales.com/rss1.xml* proporcionará acceso al *RSS*.

Servir RSS: Creación dinámica

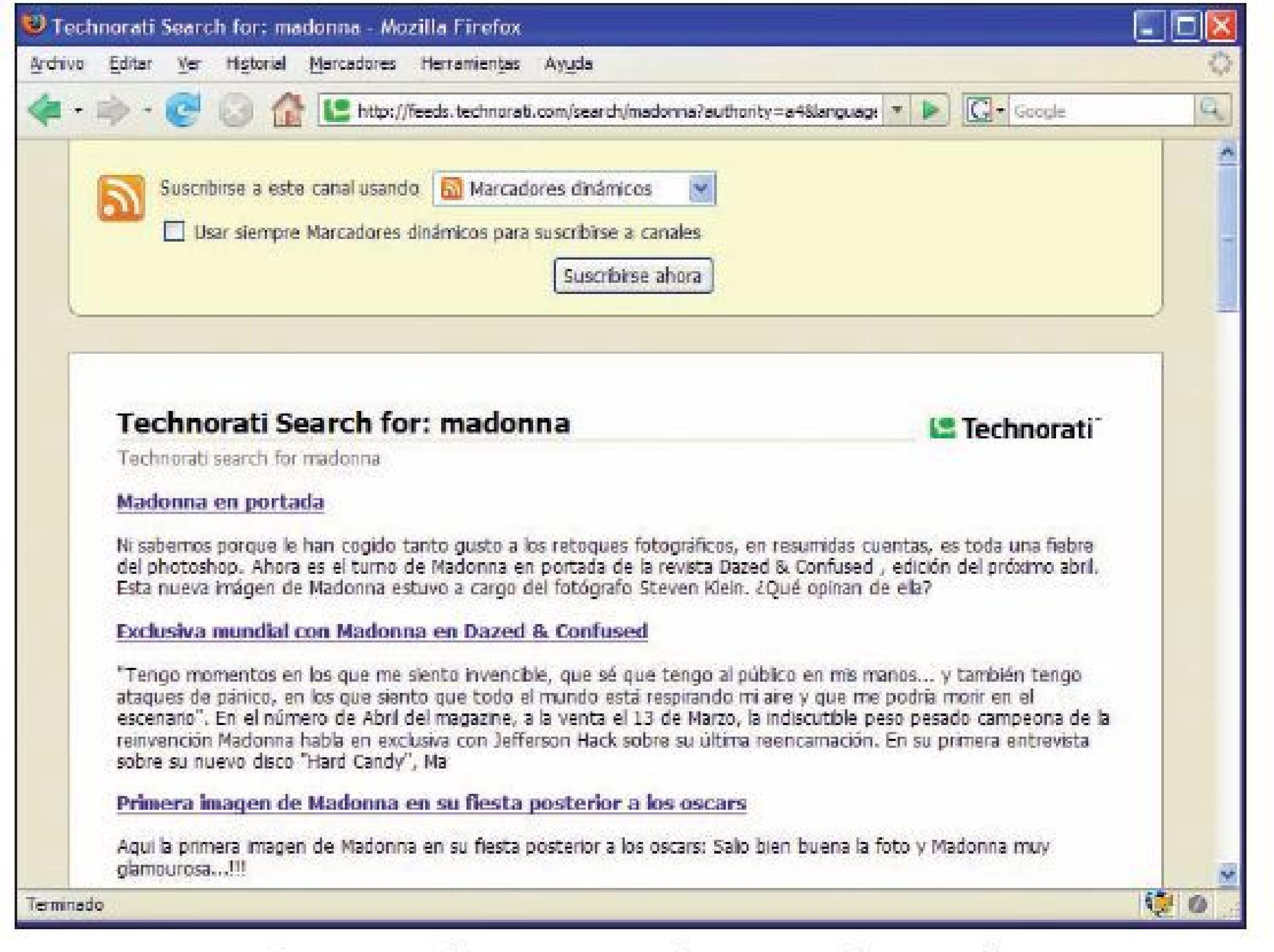
La aproximación anterior sólo tiene fines didácticos. Normalmente los *RSS* se crean dinámicamente tomando datos procedentes de bases de datos, ficheros *XML*, índices o de cualquier otro origen. El propósito de la clase *RssDocument*, perteneciente al paquete *com.spp.rss*, es precisamente proporcionar la posibilidad de construir dinámicamente un documento *RSS*. El formato elegido es *RSS 2.0*. Extender su uso para que pueda generar todos los formatos de *RSS* existentes en la actualidad es algo relativamente fácil. El ejemplo se ha ceñido a un solo formato para simplificar y mostrar de una forma clara cómo se construyen los documentos utilizando las clases estándar de *Java* para *XML*.

La clase *RssDocument* cuenta con tres atributos o propiedades:

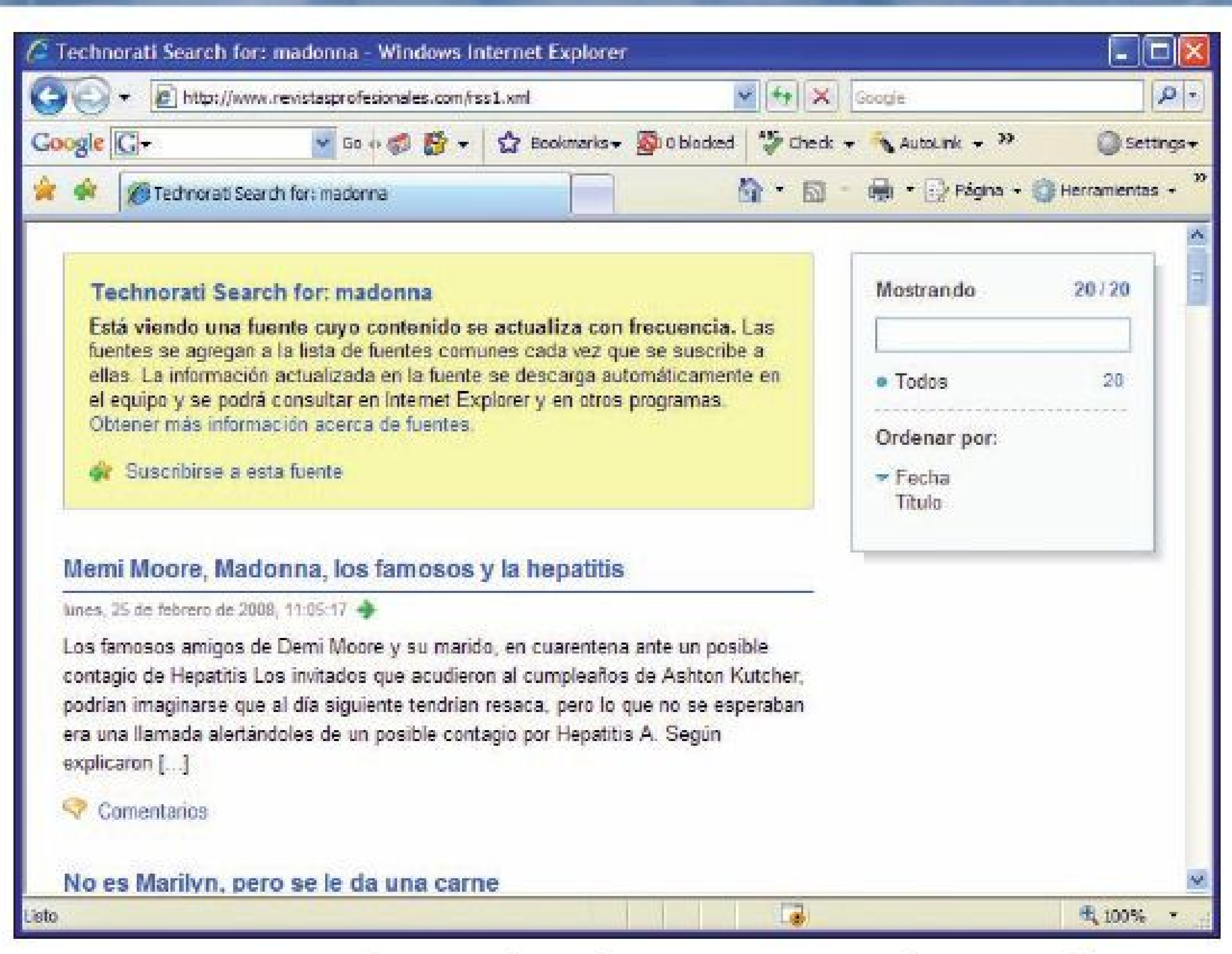
```
private final SimpleDateFormat
dateFormatterRFC822;
private final Document doc;
private final Element  eChannel;
```

El primero es un objeto de tipo *SimpleDateFormat*, una clase perteneciente al paquete estándar *java.text* del *API* de *Java*. Con esta clase se puede transformar una fecha en un formato apto para los *RSS*. En concreto el estándar que debe emplearse es el que fija el documento *RFC 822* (*www.w3.org/Protocols/rfc822/*). El segundo atributo de la clase *RssDocument* es un





Fuente RSS tal y como la muestra el navegador Firefox.



Fuente RSS generada por el servlet SRssServer1 tal y como la muestra Internet Explorer.

objeto de tipo *Document*. *Document* es una interfaz, perteneciente al paquete *org.w3c.dom* del *API* estándar de *Java*, que representa un documento *XML* en formato *DOM*. Por último el tercer atributo es un objeto de tipo *Element*, una interfaz también perteneciente al paquete *org.w3c.dom*, que representa el nodo *<channel>* dentro del *XML* correspondiente al *RSS*.

El constructor de la clase se define de la siguiente forma:

```
public RssDocument(String sChannelTitle,
String sChannelLink, String sChannel
Description)
throws ParserConfigurationException
```

Los parámetros se corresponden respectivamente con el título, el enlace y la descripción del canal.

En el código del constructor primeramente se crea la instancia de la clase *Simple DateFormat* para poder dar el formato *RFC 822* a las fechas:

```
dateFormatterRFC822 = new SimpleDate
Format(“EEE’, ‘dd’ ‘MMM’ ‘yyyy’ ‘HH:mm
:ss’ ‘Z”, Locale.US);
```

Seguidamente se prepara lo necesario para crear un objeto de tipo *Document* que represente al *XML* correspondiente al *RSS*:

```
docBuilderFactory = DocumentBuilder
Factory.newInstance();
docBuilder = docBuilderFactory.new
DocumentBuilder();
doc = docBuilder.newDocument();
```

La clase abstracta *DocumentBuilderFactory*, perteneciente al paquete *javax.xml.parsers* del *API* estándar de *Java*, cuenta con un método estático, *newInstance*, que devuelve una instancia precisamente de dicha clase. A continuación el método *newDocumentBuilder* devuelve una instancia de *DocumentBuilder*, otra clase abstracta que se incluye en el paquete *javax.xml.parsers*. Finalmente el método *newDocument* se encarga de crear un nuevo objeto *Document* con el que se puede comenzar a trabajar.

La creación dinámica del documento *XML* con los métodos del *DOM* es sencilla. La raíz es el elemento *<rss>*. El método *create Element* de la interfaz *Document* recibe como parámetro el nombre del elemento y lo crea. Después con el método *setAttribute* se establece el atributo *version* del elemento *<rss>*. Finalmente el elemento raíz del *XML* se añade al documento con el método *appendChild*:

```
Element eRss = doc.createElement(“rss”);
eRss.setAttribute(“version”, “2.0”);
doc.appendChild(eRss);
```

Las tres líneas de código anteriores resumen la mayor parte de las tareas que van a realizarse con el *DOM*: crear nodos, crear atributos y construir el árbol correspondiente al documento *XML* haciendo que unos nodos se conviertan en hijos de otros. La creación del elemento *<channel>* es todavía más sencilla ya que ni siquiera tiene atributos:

```
eChannel = doc.createElement(“channel”);
eRss.appendChild(eChannel);
```

El último paso del constructor consiste en crear los elementos obligatorios *<title>*, *<link>* y *<description>* del canal:

```
Element eTitle = doc.createElement
(“title”);
eChannel.appendChild(eTitle);
eTitle.appendChild(doc.createTextNode
(sChannelTitle));

Element eLink = doc.createElement
(“link”);
eChannel.appendChild(eLink);
eLink.appendChild(doc.createTextNode
(sChannelLink));

Element eDescription = doc.createElement
(“description”);
eChannel.appendChild(eDescription);
eDescription.appendChild(doc.create
TextNode(sChannelDescription));
```

En estos casos se ha empleado un nuevo método del *DOM* estándar: *createTextNode*. Desde el punto de vista del *DOM* las cadenas de texto que contienen los elementos son nodos, aunque de otro tipo. Se crean con ese método especial pero se trabaja con ellos del mismo modo que se hace con los nodos devueltos por *createElement*.

Añadir elementos al RSS (<item>)

El método *addItem* de la clase *RssDocument* sirve para añadir elementos *<item>* al *RSS*. El constructor recibe tres parámetros que se corresponden respectivamente con el título, enlace y descripción del elemento:

```
public final void addItem(String
sItemTitle, String sItemLink, String
sItemDescription)
```

La creación de los elementos *<item>* es casi idéntica a la creación del elemento *<channel>*, con su título, enlace y descripción. La única diferencia es que una vez creado el elemento *<item>* éste se establece como hijo del elemento *<channel>*:

```
Element eItem = doc.createElement
("item");
eChannel.appendChild(eItem);

Element eTitle = doc.createElement
("title");
eItem.appendChild(eTitle);
eTitle.appendChild(doc.createTextNode
(sItemTitle));

...
```

El elemento *<pubDate>* no es obligatorio para los elementos *<item>*. Por ello se han definido otra versión del método *addItem*:

```
public final void addItem(String
sItemTitle, String sItemLink, String
sItemDescription, Date dPubDate)
```

Cuando la fecha es distinta de *null* se crea el correspondiente nodo en el *XML* dependiendo del elemento *<item>*:

```
if (dPubDate!=null) {
  Element ePubDate = doc.createElement
  ("pubDate");
  eItem.appendChild(ePubDate);
  ePubDate.appendChild(doc.createText
  Node(dateFormatterRFC822.format(
  dPubDate)));
}
```

Añadir un elemento *<image>* al canal

El elemento *<channel>* de un *RSS* en formato *RSS 2.0* puede tener un elemento *<image>* que se emplea para definir una imagen para dicho canal. El formato en *XML* es el siguiente:

```
<image>
  <url>http://static.technorati.com/
  pix/logos/logo_sm.gif</url>
  <title>Technorati logo</title>
  <link>http://technorati.com/search/
  madonna</link>
</image>
```

La imagen es opcional por lo que no se prevé la creación de este elemento en el constructor de la clase *RssDocument* sino que se ha creado un nuevo método, *addChannelImage*, que sirve a este propósito:

```
public final void addChannelImage(
String sImageTitle, String sImageLink,
String sImageUrl)
```

Los parámetros que recibe el método se corresponden respectivamente con el título, el enlace y la *URL* de la propia imagen. La creación de todos estos elementos en el *DOM* sigue los mismos principios que se han visto anteriormente. Así el elemento *<image>* se crea con el método *createElement* y con *appendChild* se hace que sea descendiente directo del elemento *<channel>*:

```
Element eImage = doc.createElement
("image");
eChannel.appendChild(eImage);
```

El resto de los elementos de la imagen se crean del mismo modo. Por ejemplo, el elemento *<title>* de la imagen:

```
Element eTitle = doc.createElement
("title");
eImage.appendChild(eTitle);
eTitle.appendChild(doc.createTextNode
(sImageTitle));
```

Escribir el documento

El método *write* de la clase *RssDocument* es el responsable de transformar el documento *XML* en formato *DOM* representado por objeto *Document* en una cadena de caracteres. El único parámetro que acepta es un canal de escritura orientado a caracteres representado por la clase abstracta *Writer* del paquete *java.io* del *API* estándar de *Java*. Se define como sigue:

```
public final void write(Writer w)
throws TransformerFactoryConfiguration
Error, TransformerException
```

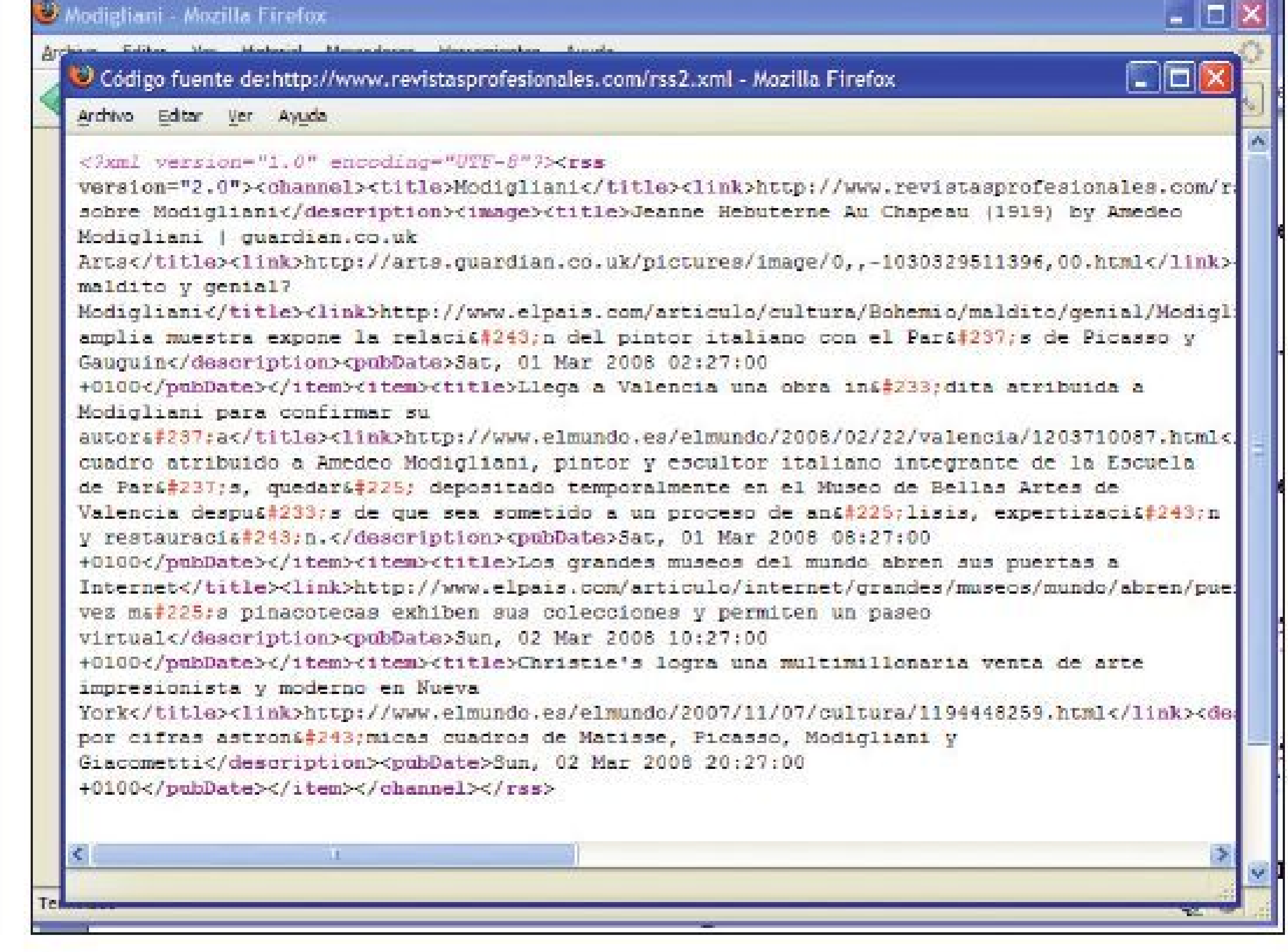
Para transformar un objeto *Document* en una cadena de caracteres, es decir, en el correspondiente código *XML*, es necesario definir en primer lugar el origen y el destino de la transformación. La clase *DOMSource*, perteneciente al paquete estándar *javax.xml.transform.Source*, se emplea para el origen. Recibe como parámetro el objeto *Document*. La clase *StreamResult*, perteneciente al paquete estándar *javax.xml.transform.Result*, define el destino de la transformación, que en este caso es simplemente un canal de escritura orientado a caracteres:

```
DOMSource domSource = new DOMSource
(doc);
StreamResult streamResult = new Stream
Result(w);
```

Definidos el origen y el destino de la transformación, hay que obtener una instancia de



Fuente RSS generada por el servlet SRssServer2 tal y como la muestra Firefox.



Código XML correspondiente a la fuente RSS generada por el servlet SRssServer2.



la clase abstracta *Transformer*, que pertenece al paquete estándar *javax.xml.transform*. Para ello se emplea el método estático *newInstance* de *TransformerFactory* y el método también estático *newTransformer* de la clas abstracta *TransformerFactory*:

```
Transformer transformer = Transformer
Factory.newInstance().newTransformer();
```

Todas estas clases se utilizan comúnmente para realizar transformaciones con *XSLT* pero como puede observarse también se emplean para obtener simplemente la cadena de caracteres correspondiente a un documento *XML* que está en formato *DOM*, representado por un objeto de tipo *Document*.

La salida producida por un objeto *Transformer* se puede configurar empleando el método *setOutputProperty*. En este caso lo único que se hace es fijar el *charset* de la cadena de caracteres resultante:

```
transformer.setOutputProperty(Output
Keys.ENCODING, "UTF-8");
```

Finalmente la transformación se lleva a cabo con el método *transform*, que recibe el origen y el destino:

```
transformer.transform(domSource, stream
Result);
```

Uso de *RssDocument*

El *servlet* llamado *SRssServer2* es la versión evolucionada del *servlet* que se mostró en el primer ejemplo. En este caso se va a generar dinámicamente el *RSS* empleando la clase *RssDocument*. Pero antes de pasar al análisis del método principal, *service*, hay que hacer una observación con respecto al uso de las clases estándar de *XML* del *API* de *Java*. *Java* proporciona una implementación de las clases abstractas e interfaces *DocumentBuilder Factory*, *DocumentBuilder*, *Document*, *TransformerFactory*, *Transformer* y otras muchas relacionadas con el tratamiento de *XML*. Ahora bien, el mecanismo de este *API* está hecho de forma que la implementación "real" se determina dinámicamente en función de las librerías que se encuentran en el *class-path* de la aplicación y considerando el valor de unas variables del sistema. La mayor parte de los servidores de aplicaciones *Web* con *Java* (*Tomcat*, *Resin*, etc.) proporcionan su propia implementación de estas clases e interfaces. Así aunque se empleen las clases e interfaces definidas en el estándar muchas veces en realidad se estarán ejecutando las implementadas por las librerías de los servidores. Una forma de controlar qué implementación es la usada consiste en establecer las propiedades que fijan precisamente dicha implementación. En el *servlet SRssServer2* se establecen estas propiedades en un bloque estático al comienzo del código:

```
static {
  try {
    System.setProperty("javax.xml.parsers.
    DocumentBuilderFactory", "com.sun.
    org.apache.xerces.internal.jaxp.Docu
    mentBuilderFactoryImpl");
    System.setProperty("javax.xml.parsers.
    SAXParserFactory", "com.sun.org.
    apache.xerces.internal.jaxp.
    SAXParserFactoryImpl");
    System.setProperty("javax.xml.
    transform. TransformerFactory",
    "com.sun.org.apache.xalan.
    internal. xsltc.trax.Transformer
    FactoryImpl");
  } catch (Exception e) {
  }
}
```

De esta forma se establece que la implementación del *DOM* es la proporcionada por el propio *API* estándar de *Java*, y en este sentido la aplicación es independiente del servidor de aplicaciones *Web* que se emplee. ¿Es esto estrictamente necesario? En realidad no lo es. En muchos casos puede ser incluso mejor dejar que las librerías del propio servidor se encarguen de dicha implementación ya que proporcionan una solución más rápida o más robusta. Pero si el objetivo es garantizar la máxima compatibilidad y portabilidad, sí resulta conveniente utilizar el estándar, y ésta es la forma de hacerlo.

Volviendo a la generación dinámica del *RSS* en el método *service* del *servlet*, primeramente se crean dos objetos, de tipo *GregorianCalendar* y *Random*, con los que se van a asignar fechas aleatorias a los elementos *item*:

```
GregorianCalendar gc = new Gregorian
Calendar(Locale.US);
Random random = new Random();
```

Evidentemente en una aplicación real estas fechas se obtendrían de algún origen de datos (una base de datos, unos ficheros *XML*, un índice, etc.) pero esta solución sirve para ilustrar el ejemplo.

La instancia de *RssDocument* se crea con los parámetros correspondientes al título, enlace y descripción del canal:

```
RssDocument rssDoc = new RssDocument
("Modigliani", "http://www.revistas
profesionales.com/rss2.xml", "Todo
sobre Modigliani");
```

Seguidamente se van añadiendo elementos *<item>* con el método *addItem* de la clase *RssDocument*. Antes de ello la fecha representada por el objeto *GregorianCalendar* creado se modifica aletatoriamente. Por ejemplo:

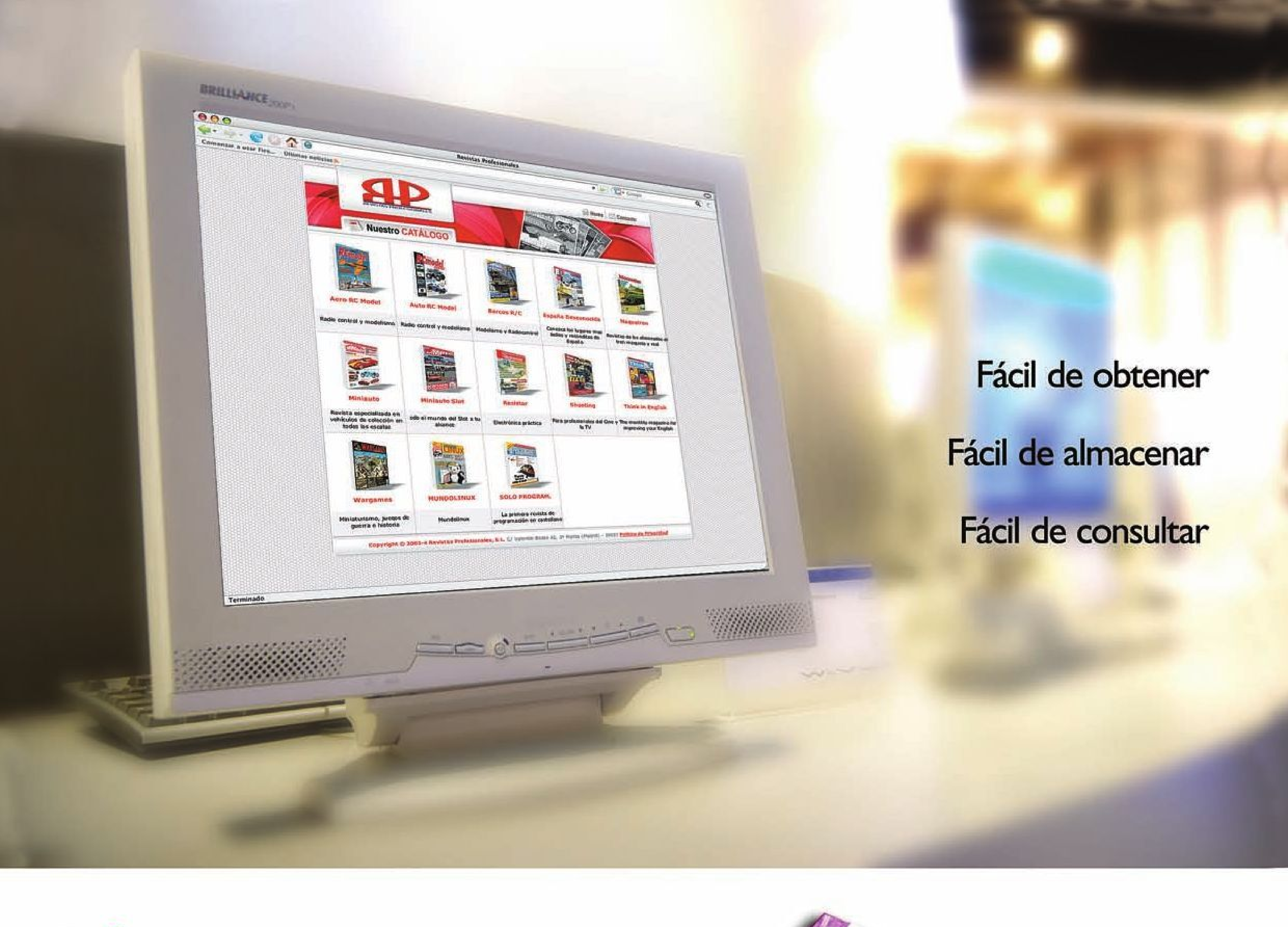
```
gc.add(Calendar.HOUR, random.nextInt
(72)-36);
rssDoc.addItem("¿Bohemio, maldito y
genial? Modigliani", "http://www.elpais.com/
articulo/cultura/Bohemio/maldito/genial/
Modigliani/elpepucul/20080124elpepucul
_20/Tes", "Una amplia muestra expone
la relación del pintor italiano con el
París de Picasso y Gauguin", gc.
getTime());
```

Después de terminar de generar todos los elementos *<item>* se configuran las cabeceras *HTTP* como en el primer ejemplo y el *servlet* genera la salida, en forma de fichero *XML*:

```
bw = new BufferedWriter(new Output
StreamWriter(res.getOutputStream(),
Charset.forName("UTF-8")));
rssDoc.write(bw);
```

Conclusión

En este primer capítulo se han visto los aspectos básicos de la creación de *RSS* con *Java* utilizando el *API* estándar. No importa si los datos proceden de una base de datos, de unos ficheros *XML* o de cualquier otro origen, ya que el proceso es siempre el mismo. Sin embargo servir *RSS* tiene algunas complicaciones adicionales que se estudiarán en las siguientes entregas. Una de las más importantes tiene que ver con el rendimiento de la aplicación. El número de usuarios potenciales que se agregan un *RSS* puede llegar a ser un muy alto. Los lectores de *RSS* hacen muchas llamadas a la *URL* que proporciona el servicio para saber si se han producido cambios y mostrárselos al usuario. Por lo tanto una aplicación *Web* que sirva *RSS* está sometida a una gran carga en general. En este sentido resulta vital controlar y optimizar la generación de los *XML*, establecer sistemas de caché eficientes y en última instancia reducir al máximo el número de *bytes* que se transfieren en cada llamada. SP


Fácil de obtener
Fácil de almacenar
Fácil de consultar



Construcción de aplicaciones con J2ME Polish

AITOR ALMEIDA ESCONDRILLAS Y DAVID SAINZ GONZALEZ

En este segundo artículo seguiremos estudiando las capacidades de J2ME Polish. Concretamente analizaremos como J2ME Polish puede ayudarnos a adaptar nuestra aplicación a diferentes dispositivos de una manera sencilla mediante el uso de ANT y directivas de preprocesado. Además veremos como mostrar mensajes de log de una manera sencilla haciendo uso del Framework de logging.

Construyendo aplicaciones con J2ME Polish

Gracias a su base de datos de dispositivos J2ME Polish facilita la personalización de una aplicación según cual sea su dispositivo objetivo. De esta forma se puede conseguir que dependiendo si la aplicación es compilada para teléfonos Nokia o Motorota se utilicen las APIs propietarias de cada fabricante. Para conseguir esto se hace uso de ANT para crear scripts que construyan la aplicación. ANT es la herramienta más utilizada a la hora de construir aplicaciones Java, ya que permite definir en ficheros XML los pasos necesarios para compilar, empaquetar y desplegar una aplicación. El elemento raiz de un fichero ANT es el elemento *project*, que a su vez contendrá elementos *target* que podrán ser llamados desde la línea de comandos. Los *target* estarán compuestos por elementos *task* que serán los que definan las acciones a llevar a cabo. Un scritp sencillo para comprobar el funcionamiento de ANT es crear un *target* que simplemente muestre un mensaje por pantalla haciendo uso del comando *echo* (ver listado 1).

También es posible definir propiedades en un fichero ANT que serán utilizadas por otros *targets*. En el listado 2 se puede ver como modificar el código del listado 1 para hacer uso de las propiedades a la hora de mostrar el mensaje.

El problema con listado 2 es que si no se ejecuta antes el *target definirMensaje* la propiedad mensaje no estará definida, esto puede resolverse haciendo uso de una propiedad *depends* (ver listado 3). De esta manera al ejecutar probar antes se ejecutará definirMensaje.

Para ver como se compilaría una aplicación, vamos a poner como ejemplo la aplicación que se puede ver en listado 4. Se trata de un midlet sencillo que muestra "Hola mundo!" en pantalla.

Para compilar este midlet, se utilizaría el ANT de listado 5. Este script de ANT se compone de varios elementos, las propiedades polish.home y wtk.home. La propiedad wtk.home no es usada en el script pero es necesaria para que J2ME Polish sepa encontrarla. *Taskdef* proporciona información sobre donde encontrar la *task j2mepolish*. *Target compilar* es el *target* que se usa para compilar el midlet. El elemento target a su vez se compone de varios elementos:

- Su propiedad *info* da información general sobre el midlet (licencia, crador, versión...) que será escrita en el *manifest*.

LISTADO 1 — Mostrar un mensaje con ANT

```
<project name="prueba" default="build">
<target name="probar">
<echo message="Esto es una prueba de ANT"/>
</target>
</project>
```

LISTADO 2 — Uso de propiedades

```
<project name="prueba" default="build">
<target name="definirMensaje">
<property name="mensaje" value="Esto es una prueba de ANT" />
</target>
<target name="probar">
<echo message=" ${mensaje}"/>
</target>
</project>
```

LISTADO 3 — Uso de depends

```
<project name="prueba" default="build">
<target name="definirMensaje">
<property name="mensaje" value="Esto es una prueba de ANT" />
</target>
<target name="probar" depends="definirMensaje">
<echo message=" ${mensaje}"/>
</target>
</project>
```



LISTADO 4 — Midlet "Hola Mundo"

```
Package prueba;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class HolaMundo extends MIDlet implements CommandListener {
    private Command exitCommand;
    private TextBox tbox;

    public HelloWorld() {
        salir = new Command("Salir", Command.EXIT, 1);
        tbox = new TextBox("Midlet prueba", " Hola Mundo!", 30, 0);
        tbox.addCommand(salir);
        tbox.setCommandListener(this);
    }

    protected void startApp() {
        Display.getDisplay(this).setCurrent(tbox);
    }

    protected void pauseApp() {}
    protected void destroyApp(boolean bool) {}

    public void commandAction(Command cmd, Displayable disp) {
        if (cmd == salir) {
            destroyApp(false);
            notifyDestroyed();
        }
    }
}
```

LISTADO 5 — Script ANT para el Midlet "Hola Mundo"

```
<project name="ejemplo" default="j2mepolish">
<property name="polish.home" location="C:\J2ME-Polish" />
<property name="wtk.home" location="C:\WTK" />
<taskdef name="j2mepolish"
classname="de.enough.polish.ant.PolishTask"
classpath="${polish.home}/import/enough-j2mepolish-build.jar:
${polish.home}/import/jdom.jar"/>
<target name="compilar">
<j2mepolish>
<info license="Apache2" name="MidletHolaMundo" vendorName="desarrollador" ver-
sion="1.0.0" jarName="prueba.jar" />
<deviceRequirements>
<requirement name="Identifier" value="Generic/midp2" />
</deviceRequirements>
<build usePolishGui="false">
<midlet class="prueba.HolaMundo" />
</build>
<emulator />
</j2mepolish>
</target>
<target name="clean">
<delete dir="build" />
<delete dir="dist" />
</target>
</project>
```

- La propiedad *jarName* indica cuál será el nombre del JAR generado.
- El elemento *deviceRequirements* define para que dispositivos se construirá la aplicación. Se pueden seleccionar los dispositivos específicos o indicar cuales deben ser las capacidades de los dispositivos, en este caso tener el perfil MIDP 2.0.
- El elemento *build* indica como compilar la aplicación. En este caso se indica cual es el midlet de la aplicación y que no se hace uso del GUI de J2ME Polish.
- El elemento *emulator* sirve para especificar que emulador se utilizará para ejecutar la aplicación.

Finalmente el *target clean* servirá para borrar compilaciones anteriores.

Cuándo se ejecuta el script de ANT se dan una serie de pasos (ver Figura 1). Primero se preprocesa el código para adaptar la aplicación a diferentes dispositivos. El preprocesado se consigue insertando instrucciones en el código (ver listado 6) que permiten que partes sólo se incluyan si el dispositivo

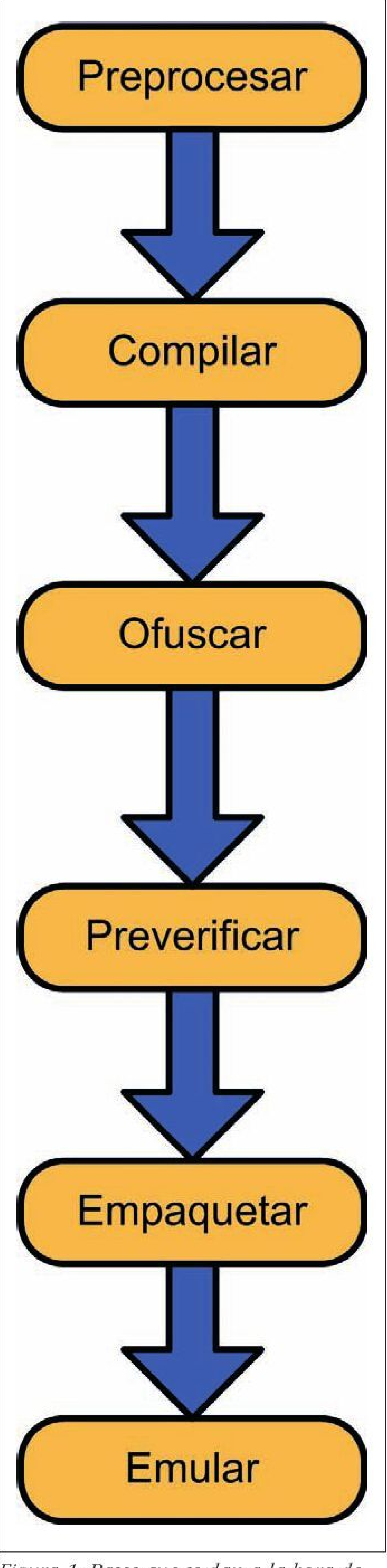


Figura 1. Pasos que se dan a la hora de ejecutar un script.



objetivo tiene las capacidades necesarias para ejecutarlo.

Una vez preprocesado el código es compilado y ofuscado. Mediante la ofuscación se consigue reducir el tamaño de la aplicación además de hacer más complicado que sea decompilada. Una vez ofuscado el código se preverifica, comprobando su integridad y su validez.

Una vez hecho esto la aplicación es empaquetada. J2ME Polish empaqueta la aplicación para cada dispositivo y localización que se especifique en el script. Se puede seleccionar que recursos incluir dependiendo de las capacidades del dispositivo. Por último la aplicación es ejecutada en el emulador que se haya especificado.

LISTADO 6 — Instrucciones de preprocesado

```
if (nuevo SMS)
{
//#if polish.midp2
try{
InputStream input = getClass().getResourceAsStream("/files/audio/nuevoSMS.mp3");
Player player = Manager.createPlayer(input, "audio/mpeg");
player.start();
} catch (IOException ex) {
  ex.printStackTrace();
} catch (MediaException ex) {
  ex.printStackTrace();
}
//#else
mostrarAlerta("Ha llegado un nuevo SMS");
//#endif
}
```

LISTADO 7 — Añadir recursos

```
<target name="j2mepolish">
<j2mepolish>
<info license="Apache2" name="MidletHolaMundo" vendorName="desarrollador" version="1.0.0" jarName="prueba.jar" />
<deviceRequirements>
<requirement name="Identifier" value="Generic/midp2" />
</deviceRequirements>
<build usePolishGui="false">
<midlet class="prueba.HolaMundo" />
<resources dir="files" excludes="pruebas*"/>
</build>
<emulator />
</j2mepolish>
</target>
```

LISTADO 8 — Añadir recursos

```
<resources dir="files">
<fileset dir="files/audio/midi" includes="*.mid" if="polish.audio.midi"/>
</resources>
```

LISTADO 9 — Ofuscar un MIDLet

```
<build>
<midlet class="prueba.HolaMundo" />
<resources dir="files" excludes="pruebas*"/>
<obfuscator name="UnOfuscador" >
<parameter name="optimize" value="false" />
</obfuscator>
<obfuscator name="OtroOfuscador" >
<parameter name="optimize" value="true" />
</obfuscator>
</build>>
```

Añadir recursos

Es habitual que la aplicación diseñada necesite ficheros extra como imágenes o audio. Por defecto los recursos a añadir en el empaquetado deben añadirse en el directorio *resources*, aunque es posible cambiar la ubicación o excluir ciertos ficheros. En el listado 7 se puede ver como incluir aquellos recursos que se encuentren en el directorio *files* excepto los ficheros que comiencen por "pruebas".

Además es posible incluir recursos específicos para los dispositivos o fabricante, creando directorios en *resources* con sus nombres. Por ejemplo, los ficheros que se encuentren en *resources/Motorola* sólo se incluirán en el empaquetado de los dispositivos de ese fabricante y los que se encuentren en *resources/Nokia/N95* sólo se incluirán en el empaquetado para el N95. También es posible definir recursos para un grupo de dispositivos. Estos grupos se encuentran definidos en la base de datos de J2ME Polish y agrupan dispositivos con características comunes tal como vimos en el anterior artículo. Por ejemplo los ficheros que se encuentren en *resources/mmapi* solo serán incluidos en los dispositivos que soporten el API multimedia. Por último es posible filtrar los recursos según las capacidades de los dispositivos. En el listado 8 se puede ver como sólo se incluyen los recursos de *files/audio/midi* en caso de que el dispositivo soporte los midis.

Ofuscar la aplicación

Como ya hemos comentado anteriormente ofuscar las aplicaciones tiene varias ventajas. Por un lado evita que el código pueda ser decompilado y copiado por otras personas, por lo que es algo totalmente necesario para las aplicaciones comerciales. Por otro lado hace las aplicaciones más compactas reduciendo su tamaño, ya que los ofuscadores eliminan las clases y métodos no utilizados. Para indicar que ofuscador utilizar habrá que hacer añadir un nodo *obfuscator* dentro del nodo *build* (ver listado 9). Si se quiere utilizar más de uno sólo habrá que añadir un nodo por ofuscador.

Firmado de MIDlets

Es posible firmar los midlets mediante el script de J2ME Polish. Las aplicaciones no firmadas son consideradas como potencialmente peligrosas por la máquina virtual



LISTADO 10 Firmado de MIDlets

```
<project name="ejemplo" default="j2mepolish">
<property name="polish.home" location="C:\J2ME-Polish" />
<property name="wtk.home" location="C:\WTK" />
<taskdef name="j2mepolish"
classname="de.enough.polish.ant.PolishTask"
classpath="${polish.home}/import/enough-j2mepolish-build.jar:
${polish.home}/import/jdom.jar"/>
<target name="compilar">
<j2mepolish>
<info license="Apache2" name="MidletHolaMundo" vendorName="desarrollador" version="1.0.0" jarName="prueba.jar" permissions=
"javax.microedition.io.Connector.http"/>

<deviceRequirements>
<requirement name="Identifier" value="Generic/midp2" />
</deviceRequirements>
<build usePolishGui="false">
<midlet class="prueba.HolaMundo" />
<sign keystore="miAlmacen.ks" key="miFirma" password="miPass" />

</build>
<emulator />
</j2mepolish>
</target>
<target name="clean">
<delete dir="build" />
<delete dir="dist" />
</target>
</project>
```

(como por ejemplo conectarse a internet) y por lo tanto esta pedirá confirmación al usuario a la hora de realizar ciertas acciones. Para poder firmar la aplicación será Nicasio generar una firma temporal mediante el programa *Keytool* del JDK. Una vez creada la firma habrá que introducir dos cambios en el script (ver listado 10). Por un lado en el nodo *info* habrá que indicar que permisos se le da al midlet mediante la propiedad *permissions*. Por otro en el nodo *build* habrá que introducir los datos de la firma generada indicando el almacén de firmas (*keystore*), las firmas (*key*) y la contraseña (*password*).

Preprocesado

El preprocesado de la aplicación sirve para indicar cuales son los requisitos para que una parte del código se incluya en una aplicación o no. Por ejemplo es posible indicar que una función de búsqueda de dispositivos mediante Bluetooth solo será llamada en caso de que el dispositivo objetivo tenga incluida la API de bluetooth. De esta manera no es necesario tener un proyecto que compilar por separado por cada dispositivo en el que se quiere que la aplicación funcione, ya que J2ME Polish se encargará de automatizarlo.

Esto se puede ver claramente en el listado 11. La instrucción de preprocesado *//#if polish.api.wma* indica que el código que viene a continuación sólo será incluido en caso que el dispositivo para el que se esta compilando la aplicación soporte la Wireless Messaging API, en caso contrario (*//#else*) se mostrará una alerta indicando que no es posible enviar el SMS. También se pueden incluir operadores lógicos en las instrucciones. El *AND* se representara con *&&*, el *OR* con ||, el *XOR* con ^ y el *NOT* con *!*.

Además es posible también hacer uso de comparadores en las instrucciones de preprocesado.

Las instrucciones que pueden utilizarse son:

- //#if: comprueba si se cumple el símbolo.
- //#else: rama alternativa de un if
- //#endif: fin de un if.
- //#ifdef: comprueba si el símbolo está definido.
- //#elifdef: rama condicional que comprueba si el símbolo está definido.
- //#ifndef: comprueba si el símbolo no está definido.
- //#elifndef: comprueba si el símbolo no está definido.

LISTADO 11 Haciendo uso de las intrucciones de preprocesado

```
public void enviarAlerta()
{
        //#if polish.api.wma
        String puerto = getAppProperty("puerto");
        String telf = getAppProperty("telefono");
//se crea la conexcon = (MessageConnection)Connector.open("sms://"+tel+":"+smsPort);
        //Se crea el mensaje
TextMessage sms=(TextMessage)con.newMessage(MessageConnection.TEXT_MESSGE);
sms.setAddress(telf + ":" + puerto);
sms.setPayloadText("Un mensaje");
//se envia
con.send(sms);
//se cierra la conexión
con.close();
//#else
crearAlerta("No se puede enviar el mensaje");
}
```

LISTADO 12 Uso de una disyuntiva en la instrucción de preprocesado

```
//#if polish.api.wma || polish.midp2
enviarSms("Una prueba");
```

LISTADO 13 Uso de comparadores

```
//#if polish.hasCamera && (polish.ScreenSize > 150x150)
ActivarCapturaVideo();
```

- //#condition: sirve para aplicar la condición a todo un fichero (ver listado 13A).
- //#define: define un símbolo extra.
- //#undefine: borra la definición de un símbolo extra.
- //#foreach: ejecuta un foreach.
- //#next: fin de un foreach.
- //#style: aplica un estilo css.
- //#include: incluye trozos de código.
- //#message: muestra mensajes en el debug.
- //#=: inserta variables de preprocesado.

La definición de variables resulta útil para poder hacer uso de ellas en el código. Por ejemplo se puede recuperar la capacidad de almacenamiento para mostrarla por pantalla (ver listado 14).

Si una propiedad contiene más de un valor puede ser recuperado uno a uno con la función *foreach*. El final de la instrucción *foreach* se marcará con *next* (ver listado 15).

Además las directivas de preprocesado pueden ser anidadas unas dentro de otra. Por ejemplo se pueden anidar condicionales si el código de una sólo debe ejecutarse después de haber ejecutado el de la otra (ver listado 16).

Para algunas propiedades es posible hacer conversiones de unidades para asegurarnos de que la comparación es adecuada. Por ejemplo si se pregunta por el tamaño de pila de video resulta conveniente transformar el valor a bytes para hacer la comparación. Para utilizar estas funciones es necesario que vayan precedidas por el símbolo de dólar y entre llaves (ver listado 17).

LISTADO 13A — Uso de condition

```
//#condition polish.api.wma
package prueba.sms;
public class EnvioDeSms {
...
}
```

LISTADO 14 — Definición de variables

```
//#ifdef polish.StorageSize
//#= int size = ${ bytes(polish.StorageSize) };
```

LISTADO 15 — Uso de foreach

```
String protocolo = "";
//#foreach protocol in polish.JavaProtocol
protocolo = "${ lowercase( protocol ) }";
procesarProtocolo(protocolo);
//#next protocol
```

Las funciones existentes son las siguientes:

- uppercase: Transforma todos los caracteres a mayusculas.
- lowercase: Transforma todos los caracteres a minusculas.
- classname: Devuelve el nombre de una clase.
- number: Devuelve el número de valores que tiene una propiedad.
- bytes: Transforma un valor a bytes. Por listado 10 kb se transformaría en 10240.
- kilobytes: Transforma un valor a kilobytes. 10240 b se transformaria en 10.
- megabytes: Transforma un valor a megabytes.
- gigabytes: Transforma un valor a gigabytes.

Estas funciones permitirán al desarrollador asegurarse de que las comprobaciones se hacen en las unidades adecuadas.

LISTADO 16 — Uso de funciones

```
//#ifdef polish.hasCamera
byte[] video = grabarVideo();
//#ifdef polish.hasVideo
mostrarVideo(video);
//#endif
//#else
System.out.println("No se disponen de capacidades de video");
//#endif
```

LISTADO 17 — Uso de funciones

```
//#ifdef polish.hasCamera
        //#if ${ bytes( polish.VideoHeapSize ) } > 20000
byte[] video = grabarVideoAltaCalidad();
//#else
byte[] video = grabarVideoBajaCalidad();
//#endif
//#ifdef polish.hasVideo
mostrarVideo(video);
//#endif
//#else
System.out.println("No se disponen de capacidades de video");
//#endif
```

Logging en J2ME Polish

El Framework de logging de J2ME Polish permite gestionar los mensajes de log de una manera sencilla mediante las directivas de preprocesado y el build.xml. Mediante el uso de estas capacidades de logging es posible solucionar dos de los problemas más comunes que se suelen dar en la gestión de los mensajes de error:

- Poder limitar la aparición de esos mensajes a las compilaciones de debug y excluirlos del producto final de una manera sencilla.
- Poder visualizar los mensajes de log de en los dispositivos reales. En el emulador es sencillo ver los *System.out.println*, pero en los dispositivos reales estos mensajes quedan ocultos. J2ME Polish proporciona un método sencillo para poder visualizarlos sin tener que desarrollar nuestra propia herramienta de logging.

Para hacer esto se hace uso de la directiva de preprocesado *#debug* y a continuación se hace un *System.out.println* con el mensaje a mostrar. Si al final del mensaje se añade una excepción el Framework de logging la reconoce automáticamente y mostrará un stacktrace. En el listado 18 se puede ver el uso de la directiva *#debug*. Los dos primeros *#debug* no tienen asignado ningún nivel de prioridad, mientras que el tercero y cuarto tienen el nivel *error*. Los niveles de prioridad son útiles para poder controlar que mensajes se muestran en todo momento cambiando el nivel de debug en el *build.xml*. Existen cinco niveles: *debug*, *info*, *warn*, *error*, y



LISTADO 18 — Uso de #debug

```
if (nuevo SMS)
{
//#if polish.midp2
try{
InputStream input = getClass().getResourceAsStream("/files/audio/nuevoSMS.mp3");
//#debug
System.out.println("Sonido cargado correctamente");
Player player = Manager.createPlayer(input, "audio/mpeg");
player.start();
//#debug
System.out.println("Sonido reproducido");
} catch (IOException ex) {
  //#debug error
  System.out.println("Error cargando fichero" + ex );
} catch (MediaException ex) {
  //#debug error
  System.out.println("Error reproduciendo sonido" + ex );
}
//#else
mostrarAlerta("Ha llegado un nuevo SMS");
//#endif
}
```

fatal. Cuando se activa un nivel de debug los niveles superiores también están incluidos. Por ejemplo si el nivel se habría situado en *info* también se mostrarían los mensajes de *warn*, *error* y *fatal*.

Además también se puede definir código que sólo se incluirá si se ha activado cierto nivel de debug. En el listado 19 se puede ver como se ha añadido estas capacidades al listado 18. Si está activado el nivel *info* se comenzara a contar cuantos SMSs llegan y se mostrará una alerta con el número de SMSs y su tiempo de llegada. Cuando se haga la compilación final este nivel de debug no estará activado y por tanto el código no será incluido en el programa.

El nivel de debug se define dentro del fichero *build.xml* haciendo uso del elemento *<debug>*. Además se pueden definir niveles de debug específicos para clases o paquetes mediante el elemento *<fitler>*. Esto se puede ver en el listado 20. Se han definido dos *targets*, *pruebas* y *final* para controlar como se desea compilar la aplicación. Los *targets* definen el valor de dos variables, la variable *pruebas* indica si se esta depurando y la variable *dir.actual* indica donde se guardará el programa compilado. En el elemento *build* se indica mediante la propiedad *workDir* donde debe guardarse las clases compiladas (esto ha sido definido en la variable *dir.actual*). Con el elemento *debug* se indica que el nivel de debug será *error* en caso de que la variable *pruebas* esté definida como *true*. Además las clases del paquete *prueba.sms* tendrán el nivel *info* y la clase *prueba.sms.Receptor* el nivel *debug*. Mediante la propiedad *showLogOnError* se indica si se quiere que J2ME Polish muestre automáticamente la ventana de log cuando se produce un error (esto sólo es posible cuando se está haciendo uso de la GUI de J2ME Polish). Cuándo la propiedad verbose está a trae J2ME Polish añade automáticamente el tiempo en milisegundos, el nombre del fichero y la línea en la que se ha producido el error al mensaje de log, facilitando de esa manera la tarea de probar el programa. En el ejemplo se ha usado la propiedad *if* para controlar cuando se activa el debug, pero también se puede hacer uso de la propiedad *unless*. Con *if* sólo se activará el debug si se produce una condición, por otra parte con *unless* se encontrará activo a menos de que se produzca una condición.

Es posible que se desee mostrar mensajes de log sin tener activado del GUI de J2ME

LISTADO 19 — Uso de directivas debug para controlar el código incluido

```
//#if polish.debug.info
int totalSMS = 0;
//#endif
...
if (nuevo SMS)
{
//#if polish.debug.info
totalSMS++;
long tiempoRecep = System.currentTimeMillis();
mostrarAlerta("El mensaje número " + totalSMS + " ha llegado en el tiempo " + tiempoRecept);
//#endif
//#if polish.midp2
try{
InputStream input = getClass().getResourceAsStream("/files/audio/nuevoSMS.mp3");
//#debug
System.out.println("Sonido cargado correctamente");
Player player = Manager.createPlayer(input, "audio/mpeg");
player.start();
//#debug
System.out.println("Sonido reproducido");
} catch (IOException ex) {
  //#debug error
  System.out.println("Error cargando fichero" + ex );
} catch (MediaException ex) {
  //#debug error
  System.out.println("Error reproduciendo sonido" + ex );
}
//#else
mostrarAlerta("Ha llegado un nuevo SMS");
//#endif
}
```



LISTADO 20 Selección del nivel de debug

```
<project name="ejemplo" default="j2mepolish">
<property name="polish.home" location="C:\J2ME-Polish" />
<property name="wtk.home" location="C:\WTK" />
<taskdef name="j2mepolish"
classname="de.enough.polish.ant.PolishTask"
classpath="${polish.home}/import/enough-j2mepolish-build.jar:
${polish.home}/import/jdom.jar"/>
<target name="pruebas">
<property name="pruebas" value="true" />
<property name="dir.actual" location="build/prueba" />
</target>
<target name="final">
<property name="pruebas" value="false" />
<property name="dir.actual" location="build/final" />
</target>
<target name="j2mepolish">
<j2mepolish>
<info license="Apache2" name="MidletHolaMundo" vendorName="desarrollador" version="1.0.0" jarName="prueba.jar" />
<deviceRequirements>
<requirement name="Identifier" value="Generic/midp2" />
</deviceRequirements>
<build usePolishGui="true" workDir="${dir.actual}">
<midlet class="prueba.HolaMundo" />
<debug showLogOnError="true" verbose="true" level="error" if="pruebas">
<filter package="prueba.sms" level="info"/>
<filter class="prueba.sms.Receptor" level="debug"/>
</debug>
</build>
<emulator />
</j2mepolish>
</target>
</project>
```

LISTADO 21 Mostrar un mensaje de log en nuestro midlet

```
Package prueba;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import de.enough.polish.util.Debug;

public class HolaMundo extends MIDlet implements CommandListener {
    private Command exitCommand;
    private TextBox tbox;


    public HelloWorld() {
        salir = new Command("Salir", Command.EXIT, 1);
        tbox = new TextBox("Midlet prueba", " Hola Mundo!", 30, 0);
        tbox.addCommand(salir);
        tbox.setCommandListener(this);
 //#ifdef polish.debugEnabled
 Debug.showLog(Display.getDisplay(this));
 //#endif

    }

    protected void startApp() {
        Display.getDisplay(this).setCurrent(tbox);
    }

    protected void pauseApp() {}
    protected void destroyApp(boolean bool) {}

    public void commandAction(Command cmd, Displayable disp) {
        if (cmd == salir) {
            destroyApp(false);
            notifyDestroyed();
        }
    }
}
```

Polish, en ese caso el Framework facilita una herramienta para mostrar mensajes de log: *de.enough.polish.util.Debug*. Haciendo uso del método *showLog(Display display)* se podrán mostrar los mensajes que se deseen en un dispositivo real. En el listado 21 se puede ver como mostrar el log. Otra opción es definir un botón para mostrar el log que sólo se añadirá en caso de que se este depurando la aplicación.

Conclusión

En este Segundo artículo hemos visto como J2ME Polish facilita la labor de personalizar las aplicaciones para diferentes tipos de dispositivos y como crear scripts de ANT para automatizar esta labor. Además hemos comprobado como las directivas de preprocesado permiten adaptar el código de nuestra aplicación a las diferentes capacidades del dispositivo y como facilitan la labor de depurar una aplicación.

En el siguiente artículo de la serie veremos como facilita J2ME Polish la creación de interfaces mediante en uso de CSS y las librerías de utilidades que ofrece.



Participa en JavaCup 2008, el torneo de fútbol virtual Java



Organizan







Patrocinan







Idea original y desarrollo del framework
JORGE RUBIRA

JavaCup 2008

Es un concurso que consiste en un torneo de fútbol virtual, basado en eliminatorias, donde cada equipo será una clase Java que implementará la estrategia del mismo, apoyándose en un framework creado para tal efecto. Para participar, sólo tienes que implementar tu equipo (una clase Java) y enviárnoslo.

Premios:
- 1500 euros para el 1º
- 1000 euros para el 2º
- 500 euros para el 3º
- 250 euros para el 4º

además de una suscripción anual a la revista *Sólo Programadores.*

http://javacup.javahispano.org





Programando en Java la Web Semántica con Jena (III)

Dr. DIEGO LZ. DE IPIÑA GZ. DE ARTAZA (Profesor del Departamento de Ingeniería del Software de la Facultad de Ingeniería de la Universidad de Deusto - ESIDE)

En esta tercera entrega exploramos las tecnologías que constituyen lo que se denomina como "web semántica con minúsculas", es decir, una simplificación práctica de los conceptos y tecnologías de la Web Semántica aplicable desde ya mismo para añadir más inteligencia a nuestros portales Web 2.0.

Introducción

Este artículo tiene por objeto analizar el conjunto de tecnologías que hacen posible la "web semántica con minúsculas", esto es, cómo acercarnos progresivamente y de una manera más sencilla hacia la todavía quimera de la Web Semántica modificando de manera incremental a través de tecnologías no intrusivas el contenido de marcado de los portales Web 2.0 actuales.

A continuación, enumeramos las diferencias entre la Web Semántica convencional, revisada y puesta en práctica en las anteriores dos entregas de esta serie y su versión simplificada "en minúsculas" atendiendo a diversos criterios, para así aclarar su distinción:

- **Filosofía.** La Web Semántica utiliza un formato de datos común para expresar el significado de los datos. Hace uso de ontologías para ayudar a las máquinas y agentes software a comprender el contenido web. Por su parte, la "web semántica" pone a los usuarios primero y a las máquinas en segundo lugar, anotando el contenido en la Web tradicional con etiquetas y atributos especiales. Es decir, no reinventa la web, simplemente la mejora para que sea más procesable por máquinas sin dejar de serlo para los humanos.
- **Lenguajes.** La Web Semántica se apoya en los lenguajes para la descripción de vocabularios semánticos RDF, RDFS y OWL, descritos en las dos anteriores entregas. La "web semántica" hace uso de nuevos lenguajes de modelización semántica que complementan XHMTL y XML como son los microformatos, RDFa y GRDDL.
- **Formato.** La Web Semántica requiere documentos RDF bien formados en formatos como RDF/XML, N3 o Turtle, mientras que la "web semántica" acepta los documentos convencionales XHTML.
- **Semántica.** En la Web Semántica, el significado es expresado por el modelo ontológico subyacente denotado en OWL. En la "web semántica" el modelado semántico no es tan elaborado y formal aunque más fácil de denotar mediante los microformatos y RDFa.
- **Ejemplos de lenguajes.** Mientras que en la Web Semántica nos encontramos vocabularios como FOAF, OWL-S y OWL-Time que modelan formalmente las relaciones entre personas, los servicios web y las relaciones temporales, en la "web semántica" nos encontramos microformatos menos exhaustivos como hCard para describir contactos, hReview para describir opiniones o la etiqueta rel para etiquetar con sentencias RDFa conceptos de una página.

Las siguientes secciones ilustran de manera práctica el conjunto de tecnologías que nos permitirán progresar de la Web 2.0 actual hacia la Web 2.0 mejorada con anotaciones semánticas o "web semántica con minúsculas": Microformatos, RDFa y GRDDL. Dejaremos hasta la próxima entrega su puesta en práctica en un proyecto real.

Microformatos

Los microformatos (a veces abreviados como ÌF o uF) añaden semántica al marcado web tradicional (HTML, CSS) para hacer que éste pase a ser de entendible únicamente por las máquinas a ser procesable por las máquinas. Es decir, pretende cambiar el modo en que interaccionamos con la web al añadir significado semántico mínimo al contenido legible por los humanos que de otra manera, desde el punto de vista de las máquinas, sería simplemente texto en un formato fácilmente renderizable pero del que difícilmente se puede extraer algún significado. En definitiva, los microformatos son un meca-

Material adicional

El material complementario puede ser descargado desde nuestra web www.revistasprofesionales.com



LISTADO 1

```
<div class="vevent">
   <a class="url" href="http://www.web2con.com/">http://www.web2con.com/</a>
   <span class="summary">Web 2.0 Conference</span>:
   <abbr class="dtstart" title="2007-10-05">October 5</abbr>-
   <abbr class="dtend" title="2007-10-20">19</abbr>,
 at the <span class="location">Argent Hotel, San Francisco, CA</span>
</div>
```

LISTADO 2 — Descripción en hCard

```
<div class="vcard">
 <a class="url fn" href="http://paginaspersonales.deusto.es/dipina">Diego López de Ipiña</a>
 <div class="org">Universidad de Deusto</div>
 <div class="adr">
  <div class="street-address">Avda. Universidades 24</div>
  <span class="locality">Bilbao</span>
  <span class="postal-code">48007</span>
  <span class="country-name">Spain</span>
 </div>
 <div class="tel">944139000</div>
</div>
```

nismo para introducir significado dentro de otros formatos ampliamente utilizados como HTML/XHTML o Atom/RSS.

Los microformatos permiten que los elementos de datos (eventos, contactos o localizaciones) en páginas web HTML (o XHTML) puedan ser detectados y su contenido extraído por software e indexado, buscado y referenciado de modo que pueda ser reutilizado y combinado. La web http://microformats.org documenta las extensiones semánticas a XHTML más populares tales como hCard, para describir personas y organizaciones, hCalendar, para describir calendarios y eventos, hReview, para recoger opiniones, XFN, para describir relaciones sociales o geo, para describir localizaciones geodésicas. A modo de ejemplo, el siguiente fragmento de XHTML añade anotaciones en microformato geo a un elemento HTML span para codificar la longitud y latitud del lugar descrito en una web.

```
<span class="geo"><span class="latitude">
52.48</span>, <span class="longitude">
-1.89</span></span>
```

XHTML y HTML permiten encapsular semántica dentro de los atributos de cualquier etiqueta de marcado. Los microformatos se aprovechan de estos estándares para indicar la presencia de metadatos mediante los atributos: class, rel y rev.

Ventajas y desventajas

Las principales ventajas de los microformatos son:

- Se empotran en XHTML, un lenguaje ampliamente aceptado, lo cual hace que su adopción sea más sencilla que si requiriera introducir un lenguaje semántico nuevo, tal es el caso de RDF. Eso explica la afirmación comúnmente asociado a los microformatos: "están diseñados en primer lugar para los humanos y en segundo lugar para las máquinas".
- Los fragmentos de los microformatos pueden ser alojados directamente dentro de páginas XHTML, no requieren hacer referencia a ficheros externos.
- Las herramientas actuales trabajan sin problema alguno (navegadores), ignorando los microformatos en caso de no saber qué hacer con ellos. Además, algo importante es que las futuras versiones de los mismos han planificado y anunciado soportar la identificación y tratamiento de microformatos (Firefox 3.0 e Internet Explorer 8.0).
- Se puede añadir funcionalidad adicional a la semántica actual de las etiquetas a través de los atributos class, rel y rev.

Sin embargo, obviamente no es oro todo lo que brilla. Los microformatos también presentan algunos problemas. El principal es que muchos autores distorsionan los formatos de etiquetado que alojan (XHMTL, RSS) los microformatos con nuevos atributos y etiquetas en vez de simplemente apoyarse en los atributos y elementos ya existentes. Este es el caso de XOXO.

Microformatos más utilizados

Existen varios microformatos orientados a dominios verticales muy concretos. A continuación enumeramos los más utilizados:

- XFN (http://gmpg.org/xfn) define un conjunto de valores que describen relaciones humanas (hace uso del atributo HTML rel; este mismo atributo se utiliza para extender metadatos de Creative Commons). Por ejemplo: Jane. En este caso se está indicando que Jane es una chica guapa a la que conoce y con la que tuvo una cita el autor de la página que ha dejado tal fragmento XHMTL en su página.
- Geo (http://microformats.org/wiki/geo) permite indicar detalles de la localización de un portal. A menudo aparece en las etiquetas <meta> de la cabecera. Por ejemplo: <div class="geo">GEO: 37.386013, -122.082932 </div>.
- hCalendar (http://microformats.org/wiki/hcalendar) permite indicar eventos de calendario. Mapea el formato estándar iCalendar a XHTML. Por ejemplo (ver listado 1).
- XOXO (http://microformats.org/wiki/xoxo) sirve para indicar tablas de contenidos y listas de blogs en los que uno está interesado.
- hCard (http://microformats.org/wiki/hcard) es útil para libretas de direcciones (mapea el formato libre vCard a XHTML).
- Otros microformatos de interés son: a) Reviews (http://microformats.org/wiki/hreview) – permite indicar críticas de cierto contenido, b) Resumes (http://microformats.org/wiki/hresume) – permite indicar tu CV en microformato o c) incluso microformatos para describir conceptos de vino como hwine.

No solamente están estandarizados los microformatos más comunes sino que existen varias herramientas que permiten generar fragmentos XHTML correspondientes a ellos tal es el caso del generador hCard, disponible en (http://tantek.com/microformats/hcard-creator.html). El listado 2 muestra los detalles de contacto en formato hCard correspondientes al autor de este artículo.

Uso de los microformatos

Los microformatos están ya cambiando el modo de interactuar con la web. Algunos ejemplos de sus usos son:

- Sites de agregación: la combinación de blogs y microformatos es muy interesante. Si quieres vender algo, etiquétalo con el microformato hlisting (http://microformats.org/wiki/hlisting) y portales como http://www.edgeio.com/ lo encontrarán cuando agregen anuncios clasificados por la web.
- Compartición de información con una comunidad particular: añade RSS complementado por microformato geo para las localizaciones de una pista de bicis mountain bike, y otra gente se p a la fuente usando Google Earth.



Figura 1. Concepto de navegador como un bróker de información.

- Búsqueda dirigida: mediante la publicación de un cómic acompañado de un microformato acordado por la comunidad del cómic, el resto de la comunidad podría fácilmente buscarlo usando una motor de búsqueda.
- El navegador web como un bróker de información: los navegadores web futuros asociarán datos marcados semánticamente con ciertas aplicaciones. Este concepto es ilustrado en la figura 1. Por ejemplo:
  - La información de contacto en un portal será asociada con la aplicación de contactos.
  - Los eventos serán asociados con tu aplicación de calendario favorita.
  - Las localizaciones serán asociadas con tu aplicación de mapas favorita.
  - Los números de teléfono serán asociados con tu aplicación de VoIP favorita.

Algunos portales que ya usan hoy en día microformatos son:

- Flickr – permite añadir etiquetas de geotagging a sus fotos.
- Yahoo! Local – codifica cada resultado de la búsqueda con un hCard.
- Upcoming.org – codifica eventos con hCalendar.

Herramientas de procesado de microformatos

Pronto nuestros navegadores no solamente detectarán la presencia de contenido RSS en un portal visitado si no que aparecerá un nuevo icono por cada tipo de dato estructurado en la Web. Los navegadores web deben proveer al usuario con una interfaz limpia, consistente y simple al usuario. Como ya hemos comentado las nuevas versiones de Firefox (3.0) e Internet Explorer (8.0) soportarán microformatos.

De esta manera los navegadores actuales se transformarán en brokers de información (ver figura 1) detectando la información contenida en las páginas web y pasando tal información a otras aplicaciones. Pasarán de ser renderizadores de HTML a brokers de información. Las aplicaciones de gestión de contactos, calendarios y mapas podrán utilizar la API del navegador para integrarse con el mecanismo de detección de microformatos del navegador.

Hoy en día, podemos ya empezar a experimentar con la detección y tratamiento de microformatos a través de la extensión Operator para Mozilla Firefox. La figura 2 muestra los microformatos identificados mediante la extensión Firefox Operator (https://addons.mozilla.org/es-ES/firefox/addon/4106) al acceder a la página http://www.microformats.org.

RDFa

RDF es un formato flexible para guardar, agregar y consultar metadatos. Sin embargo, la sintaxis RDF/XML es compleja provocando que mucha gente haya evitado su uso. Por tanto, es necesario encontrar una solución que permita aprovecharnos de las ventajas de RDF pero con un nivel de dificultad inferior. Tal solución es RDFa (http://www. w3.org/TR/xhtml-rdfa-primer/), una tecnología clave para la incorporación de tripletas RDF dentro de HTML, requisito imprescindible para progresar en la construcción de la Web Semántica.

El principal objetivo de RDFa es hacer sencilla la encapsulación de RDF en contenido XHMTL y XML. Añade metadatos a XHTML sin afectar lo que se ve en el navegador. De ese modo, los datos de una página web no son sólo visibles por humanos sino también tienen más sentido por procesos automatizados. Permite la agregación de datos y asociación de metadatos para realizar tareas mucho más sofisticadas que aquellas permitidas hasta el momento utilizando técnicas como "screen scrapping", es decir, obteniendo datos a partir del procesado del contenido de páginas. Utiliza algunos atributos ya existentes en XHTML 1 y algunos otros nuevos de XHTML 2 para guardar los sujetos, predicados y objetos de las tripletas RDF. Los atributos de XHMTL 1 utilizados son href, content, rel, rev y datatype. Los nuevos atributos de metainformación de XHTML 2 utilizados son about, role y property. Siempre que se genera HTML automáticamente hay una oportunidad para incorporar anotaciones en formato RDFa. De este modo, los portales donde se publican horarios de películas, listas de precios y otras muchas páginas donde se obtiene información de un sistema de bases de datos back-end, son tierra fértil para la creación de RDFa.

En RDFa como en RDF hay dos principales casos de modelado de información: a) triple-

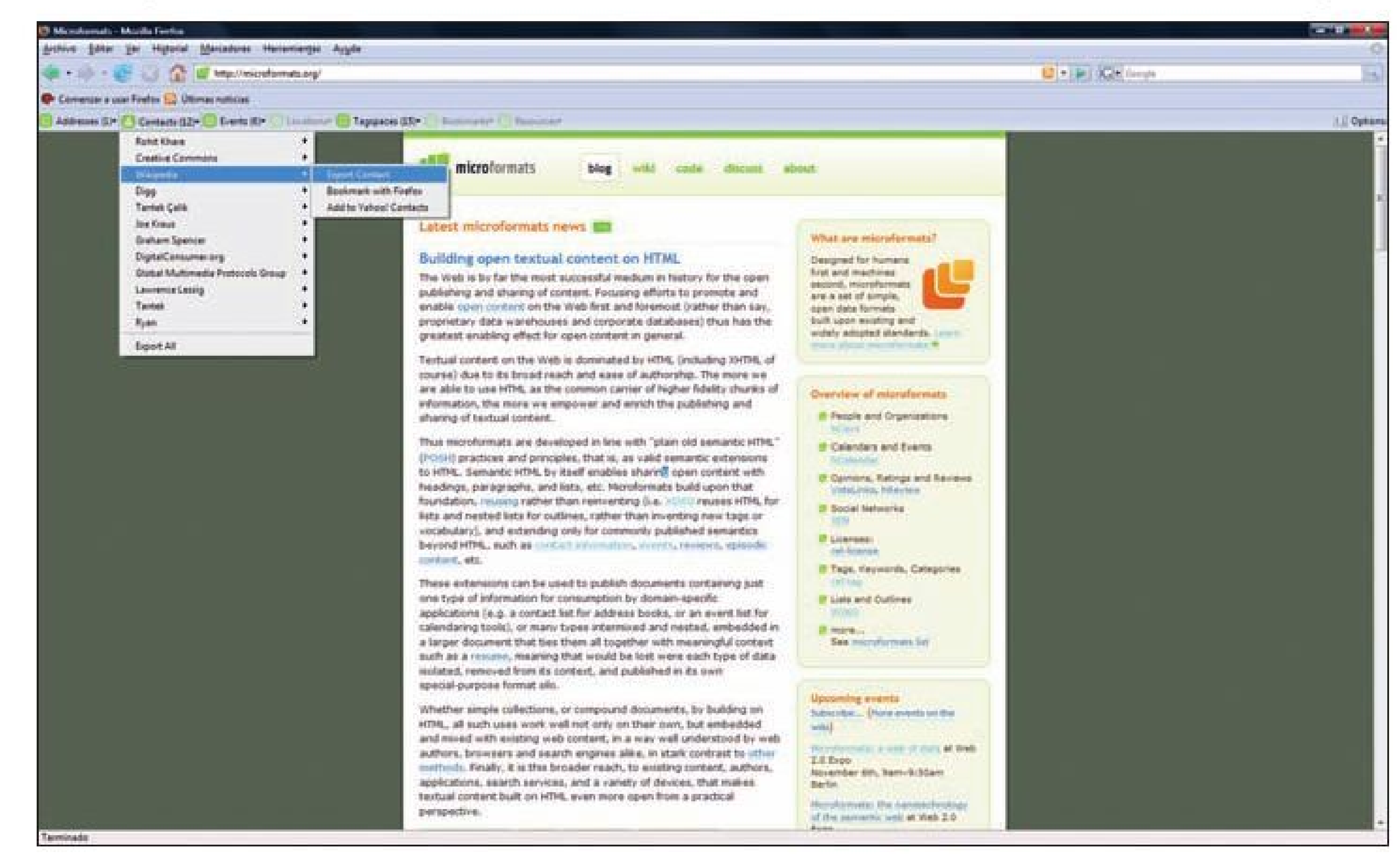


Figura 2. Firefox Operator accediendo a www.microformats.org.



TABLA 1: Reglas sintácticas de RDFa

	Sujeto	Predicado	Objeto
String como objeto	about	property	Atributo content o PCDATA
URI como objeto	about	rel	href

LISTADO 3

```
<p>Última revision del documento: <span about="
http://en.wikipedia.org/wiki/RDFa" property="dc:date"
content="20080304T15:32:00">4 de Marzo del 2008, 3:32 PM</span></p>
```

LISTADO 4

```
<rdf:Description rdf:about=" http://en.wikipedia.org/wiki/RDFa ">
<dc:date>20080304T15:32:00</dc:date> </rdf:Description>
```

LISTADO 5

```
<html xmlns:fm="http://www.foomagazine.com/ns/prod/">
  <head>
    <title>Is Black the New Black?</title>
    <meta property="fm:newsstandDate" content="2006-04-03"/>
    <meta property="fm:copyEditor" content="RSelavy"/>
    <meta property="fm:copyEdited" content="2006-03-28T10:33:00"/>
  </head>
  <body>
```

tas que tienen un literal como objeto y b) tripletas que tienen una URI como objeto. Esta segunda técnica permite reutilizar el mismo objeto en varias tripletas. Por ejemplo la siguiente sentencia XHTML combinada con RDFa indica que el site http://rdfa.info/ tiene por título según el vocabulario Dublin Core el valor "RDFa: Interoperable Web Metadata".

```
<span about="http://rdfa.info" property=
"dc:title" content=" RDFa:
Interoperable Web Metadata"/>
```

Alternativamente, esta misma sentencia podría haberse expresado de una manera aún más compacta como:

```
<span about="http://rdfa.info" property=
"dc:title"> RDFa: Interoperable Web
Metadata <span/>
```

Finalmente, la misma sentencia apuntando a una URL quedaría como:

```
<span about="http://rdfa.info" rel=
"dc:subject" href="http://www.w3.org/
TR/xhtml-rdfa-primer/"/>
```

La tabla 1 muestra la sintaxis XHTML utilizada para denotar tanto tripletas que tienen un string como una URI por objeto.

Modos de uso RDFa

Los agentes software conocedores de RDFa pueden extraer datos útiles de los elementos span, link y meta con pequeñas modificaciones sobre los mismos:

- Los elementos span son los más comúnmente utilizados en RDFa porque puedes insertarlos en cualquier lugar del cuerpo de un documento HTML, los mismos atributos RDFa pueden ser añadidos a cualquier elemento que desees.
- Los elementos link y meta se utilizan para insertar RDFa en la cabeza de un documento HTML
- El elemento de hipervínculo <a> también es popular para guardar metadatos RDFa. Expresa una relación entre un recurso (el documento donde aparece tal hiperenlace) y otro documento (el recurso al que apunta). El atributo rel de tal elemento indica la relación en la tripleta.

A continuación vamos a describir tres modos diferentes de utilizar RDFa:

- *Metadatos inline sobre componentes del documento.* Así el siguiente fragmento de código puede cualificar semánticamente la descripción de un fragmento del documento XHTML haciendo referencia a un artículo sobre RDFa (Listado 3).
- Un extractor RDFa obtendría el siguiente contenido de tal fragmento XHTML+RDFa (Listado 4).
- *Metadatos sobre el documento contenido.* Metadatos sobre el documento sin contenido visible pueden ser guardados en el elemento head del documento (Listado 5).
- Un extractor RDFa obtendría el siguiente contenido RDF en formato RDF/XML (Listado 6).
- *Metadatos fuera de banda.* De más elegante declaración separan la definición de contenido RDFa declarado en la sección head del documento de los lugares en los que es asociado (Listado 7).
- La tripleta resultante utiliza el valor de contenido de la fecha (Listado 8).

El patrón de uso de cualquier aplicación que quiera utilizar RDFa requiere 3 pasos: a) extraer las tripletas RDFa empotradas, b) cargar las tripletas en una memoria de tripletas o disco y c) desarrollar una aplicación alrededor de los datos extraídos. Existen herramientas como el servicio web publicado en http://torrez.us/services/rdfa/ que pueden ser utilizados desde nuestro código para extraer contenido RDF a partir de código XHTML+RDFa. Si quieres utilizar tal servicio web desde tus aplicaciones web semánticas solamente tienes que apuntar tu consulta SPARQL a la siguiente dirección: http:// torrez.us/services/rdfa/[URL de la página

LISTADO 6

```
<rdf:Description rdf:about="">
  <fm:newsstandDate>2006-04-03</fm:newsstandDate>
  <fm:copyEditor>RSelavy</fm:copyEditor>
  <fm:copyEdited>2006-03-28T10:33:00</fm:copyEdited>
</rdf:Description>
```

LISTADO 7

```
<html xmlns:fm="http://www.comida.com/">
<head><meta about="#recipe13941"><meta
property="fm:ComponentID">XZ3214</meta><meta
property="fm:ComponentType">Receta</meta><meta
property="fm:RecipeID">r003423</meta></meta>
</head>
<body>
  <h>Receta de Bacalao Pil Pil</h>
  <section id="recipe22143">
    <h>Perejil</h>
  </section>
  <section id="recipe13941">
    <h>Bacalao</h>
  </section>
</body>
</html>
```



XHTML que contiene código RDFa]. En local, sin necesidad de recurrir a un recurso remoto, se puede hacer uso de hojas de transformación XSLT tal como la disponible en http://www-sop.inria.fr/acacia/soft/ RDFa2RDFXML.xsl.

Características avanzadas de RDFa

Una característica destacable de RDFa es que permite construir múltiples tripletas del mismo sujeto reduciendo al máximo el código introducido. Los elementos hijo heredan el atributo about de un elemento padre. Si no se encuentra el atributo about simplemente significa que estamos cualificando el documento actual. Por ejemplo, el siguiente elemento span indica que está describiéndose un recurso correspondiente a una foto de la Alhambra creada por Diego Ipiña y de formato jpeg, es decir, tres sentencias RDF compartiendo un mismo sujeto (listado 9). Además, a través de RDFa es perfectamente posible asociar tipos a los valores de los predicados. El siguiente ejemplo XHTML+RDFa muestra cómo producir una terna RDF donde el valor asociado al predicado solop:fecha de un artículo solop:jena3 es de tipo XML Schema date (listado 10).

Como ya se ha comentado, RDFa utiliza a menudo el atributo rel como predicado de una tripleta con el atributo about para nombrar el sujeto y href para nombrar el objeto. Por el contrario, el atributo rev sirve para denotar una tripleta donde href hace referencia al sujeto y about el valor del atributo. En el siguiente ejemplo podemos apreciar como el sujeto puede convertirse en objeto dependiendo de si consideramos el predicado dado por rel o rev. Esto es de un golpe estamos generando dos sentencias RDF complementarias (listado 11).

LISTADO 8

```
<rdf:Description rdf:about="file://C|/dat/xml/rdf/rdfa/test5.html#recipe13941">
  <fm:ComponentType>Receta</fm:ComponentType>
  <fm:RecipeID>r003423</fm:RecipeID>
  <fm:ComponentID>XZ3214</fm:ComponentID>
</rdf:Description>
```

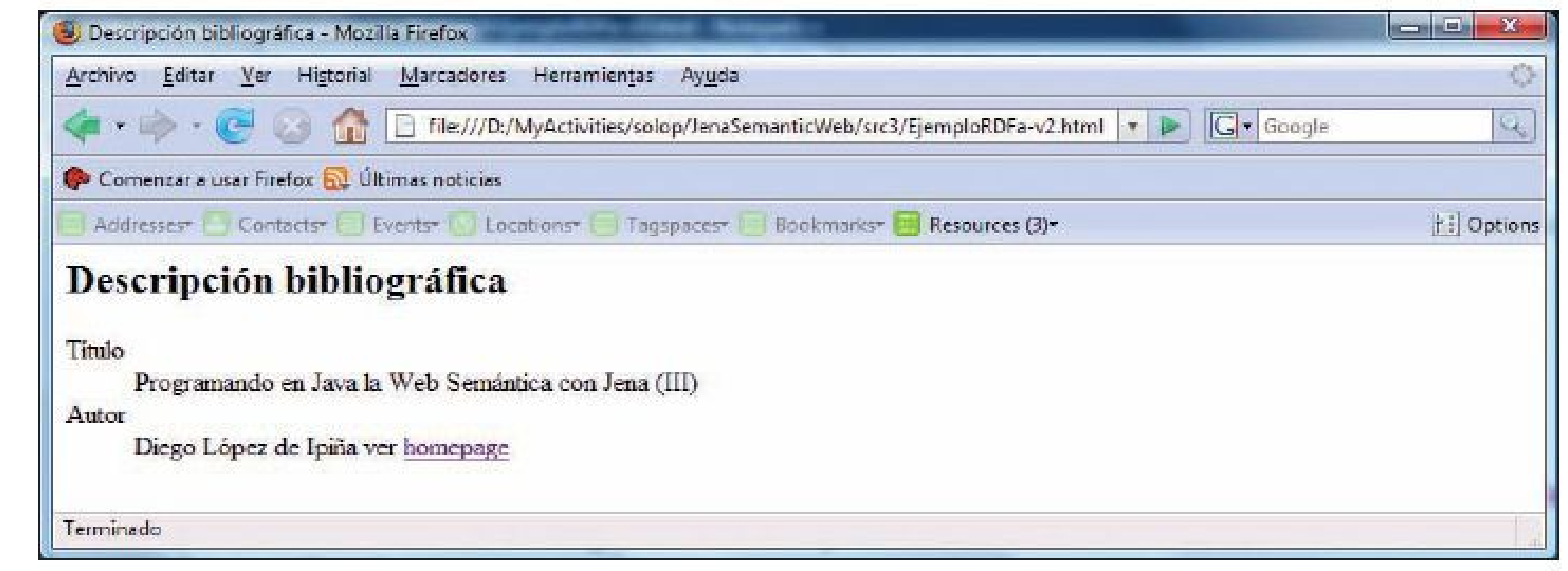


Figura 3. Página XHTML+RDFa mostrada en Firefox.

LISTADO 9

```
<img src="http://www.snee.com/img/myfile.jpg" about=" http://www.flickr.com/photos/61381772@N00/33654792/"> <span property="dc:subject" content="Alhambra de Granada"/>
        <span property="dc:creator" content="Diego Ipiña"/>
        <span property="dc:format" content="img/jpeg"/>
</img>
```

LISTADO 10

```
<tr about="[solop:jena3]"><td><span property="solop:fecha" datatype=" http://www.w3.org/2001/XMLSchema#date">2008-03-10</span></td></tr>
```

LISTADO 11

```
<span about="http://paginaspersonales.deusto.es/dipina" rel="solop:writes" rev="solop:isWrittenBy" href="http://www.revistasprofesionales.com/solop/jena3"/>
```

Ejemplo completo de RDFa

El listado 12 muestra un fragmento de XHTML representando información sobre este artículo. El usuario vería en su navegador lo mostrado en la figura 3. Sin embargo, cualquier programa conocedor de RDFa podría aplicar una hoja de transformación XSLT para extraer el código RDF/XML incrustado en la página cuya representación más concisa en formato N3 es mostrada en el listado 13 y representada gráficamente en la figura 4.

LISTADO 12 — Página HTML con contenido RDFa

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet href="http://www-sop.inria.fr/acacia/soft/RDFa2RDFXML.xsl" type="text/xsl" ?>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:foaf="http://xmlns.com/foaf/0.1/" >
  <head profile="http://www.w3.org/2003/g/data-view" >
    <link rel="transformation" href="http://www-sop.inria.fr/acacia/soft/RDFa2RDFXML.xsl"/>
    <title>Descripción bibliográfica</title>
  </head>
  <body>
    <h2>Descripción bibliográfica</h2>
    <dl about="http://www.revistasprofesionales.com/solop/art20080310-6/">
      <dt>Título</dt>
      <dd property="dc:title">Programando en Java la Web Semántica con Jena (III)</dd>
      <dt>Autor</dt>
      <dd rel="dc:creator" href="#a1"> <span about="#a1">
        <link rel="rdf:type" href="[foaf:Person]" />
          <span property="foaf:name">Diego López de Ipiña</span> ver <a rel="foaf:homepage" href="http://paginaspersonales.
           deusto.es/dipina">homepage</a></span>
      </dd>
    </dl>
  </body>
</html>
```



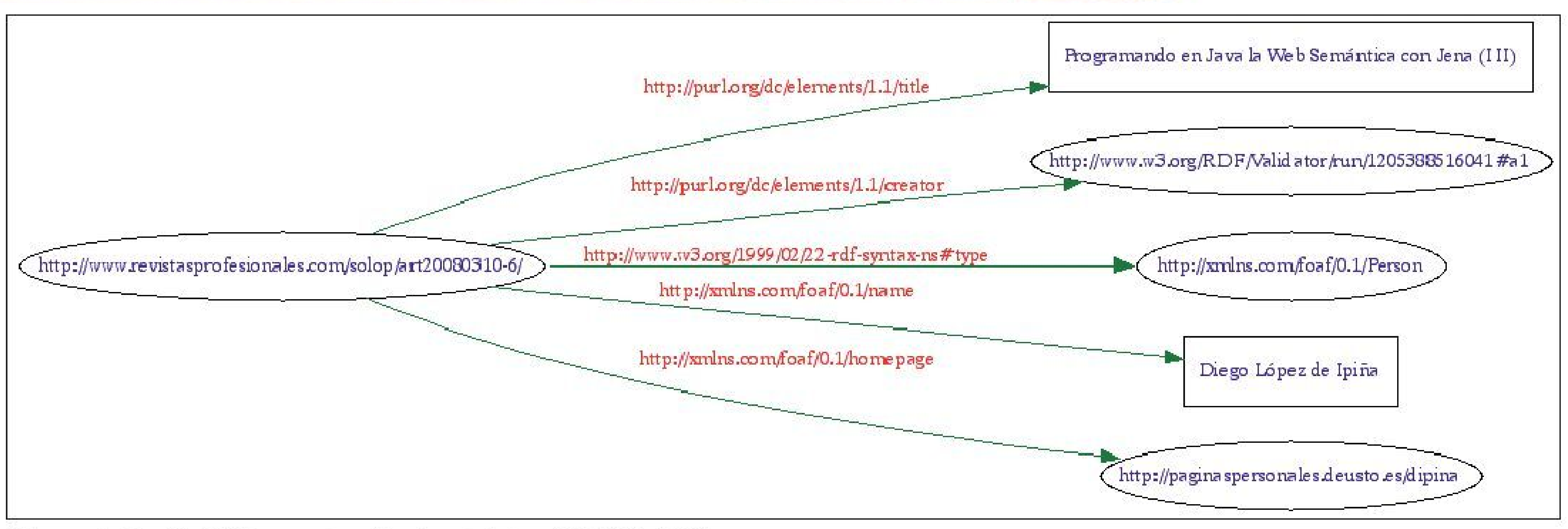


Figura 4. Grafo RDF correspondiente a página XHMTL+RDFa.

LISTADO 13 RDF extraído del listado 12 representado en formato N3

```
@prefix h: <http://www.w3.org/1999/xhtml> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xmlns: <http://www.w3.org/2000/xmlns/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
<http://www.revistasprofesionales.com/solop/art20080310-6/> dc:title "Programando en Java la Web Sem\u00E1ntica con Jena (III)" ;
        dc:creator :a1 ;
        rdf:type foaf:Person ;
        foaf:name "Diego L\u00F3pez de Ipi\u00F1a" ;
        foaf:homepage <http://paginaspersonales.deusto.es/dipina>
```

Combinando los microformatos y RDFa mediante GRDDL

GRDDL (Gleaning Resource Descriptions from Dialects of Languages) es un mecanismo para combinar eficientemente descripciones semánticas de recursos realizadas sobre diferentes dialectos de anotación semántica como los microformatos, RDFa, N3, RDFa o Embedded RDF. Define un estándar para declarar que una página web o XML puede ser transformada en un grafo RDF, así como los algoritmos o mecanismos, principalmente hojas de transformación XSLT, que permitan efectuar tales transformaciones. Mediante GRDDL podemos hacer que descripciones tanto en forma de microformatos como RDFa converjan en un documento RDF. En definitiva, GRDDL añade marcado especial a nuestras páginas XHTML y XML con un doble propósito:

- Declarar que un documento contiene fragmentos de información extraíbles (atributo profile del elemento head) y
- Asociar un algoritmo (típicamente una transformación XSLT) que realice la extracción de los datos RDF del documento (atributo transformation del elemento link).

En la siguiente URL pueden encontrarse diferentes implementaciones de motores GRDDL: http://esw.w3.org/topic/GrddlImplementations.

Ejemplo de uso de GRDDL

Consideremos el siguiente ejemplo, tomado del documento GRDDL Primer (http://www.w3.org/TR/grddl-primer/), que nos va a permitir ver claramente la utilidad del estándar GRDDL.

"Jane quiere comprobar si en algún momento el próximo año puede planificar un encuentro con dos de sus amigos en algún lugar del mundo donde coincidan, a pesar de que cada uno de ellos publica su calendario en un formato semántico diferente (microformato hCalendar, Embedded RDF y RDFa)".

La figura 5 corresponde con el proceso que el agente software a desarrollar por Jane ha de efectuar para combinar la agenda publicada por su amigo Robin en XHTML+hCalendar, por su amigo David en XHTML+Embedded RDF y por ella misma en XHTML+RDFa, para obtener un grafo RDF común sobre el que se puede efectuar una consulta SPARQL para determinar la coincidencia de estas tres personas en un evento común durante el próximo año.

El listado 14 muestra el contenido de una página web conteniendo fragmentos del microformato hCalendar correspondientes a la planificación de eventos publicados por Robin, amigo de Jane, en la URL http://www.w3.org/TR/grddl-primer/robin-hcal-grddl.html. Para poder asociar la información en este documento a un modelo de datos RDF, se han tenido que efectuar dos modificaciones clave en tal documento:

Añadir un atributo profile al elemento head para indicar que el documento contiene metadatos GRDDL: <head profile="http://www.w3.org/2003/g/data-view">

Añadir un elemento link conteniendo una referencia a la transformación GRDDL específica que convierte contenido HTML anotado con expresiones en microformato hCalendar a RDF: <link rel="transformation" href="http://www.w3.org/2002/12/cal/glean-hcal"/>

La figura 6 ilustra el proceso de transformación efectuado sobre los datos de calendario publicados por Robin. La URI del profile del documento indica al receptor que procese aquellos elementos link cuyo atributo rel es igual a transformation para extraer así los datos como RDF del calendario de Robin.

El listado 15 muestra un fragmento de código en Embedded RDF (http://research.talis.com/2005/erdf/wiki/Main/RdfInHtml), otro formato similar a RDFa que permite encapsular contenido RDF en un documento XHTML, correspondiente a los eventos planificados por David, otro amigo de Jane, publicados en http://www.w3.org/TR/grddl-primer/david-erdf.html. Como detalle reseñable, en este caso el atributo profile no contiene una referencia al perfil GRDDL. Ahora, proporciona la URI al profile estándar para Embedded RDF que a su vez contiene los



LISTADO 14 — Calendario de Robin en XHTML+hCalendar

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head profile="http://www.w3.org/2003/g/data-view">
    <title>Robin's Schedule</title>
    <link rel="transformation" href="http://www.w3.org/2002/12/cal/glean-hcal"/>
  </head>
  <body>
    <ol class="schedule">
      ...
      <li>2007
        <ol>
          <li class="vevent">
            <strong class="summary">Web Design Conference</strong> in
            <span class="location">Edinburgh, UK</span>:
            <abbr class="dtstart" title="2007-01-08">Jan 8</abbr> to
            <abbr class="dtend" title="2007-01-11">10</abbr>
          </li>
          ...
        </ol>
      </li>
    </ol>
  </body>
</html>
```

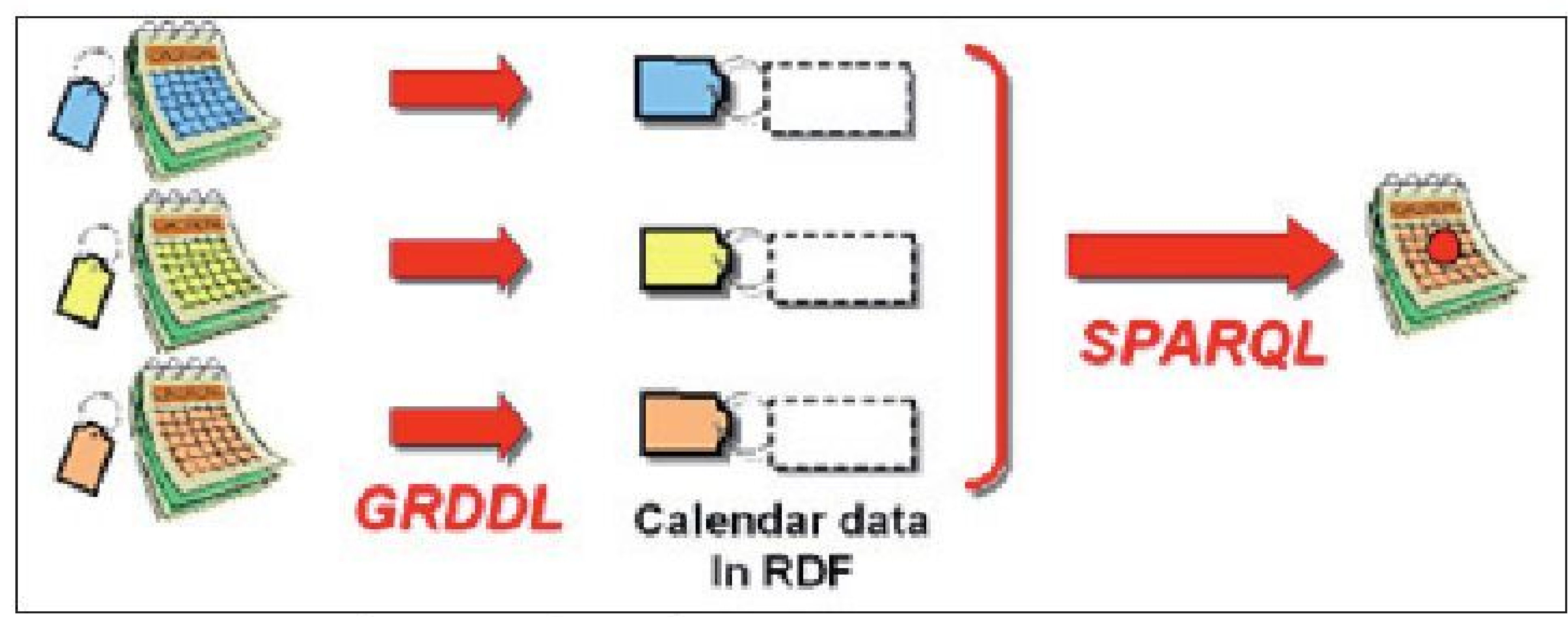


Figura 5. Proceso de agregación mediante GRDDL.

metadatos de la transformación GRDDL. Es decir, de manera indirecta hace referencia a la hoja de transformación de Embedded RDF a RDF.

Finalmente, la propia Jane publica su calendario directamente en RDFa en la siguiente dirección http://www.w3.org/TR/grddl-primer/janeschedule.html. El listado 16 muestra un fragmento de tal documento que incluye una referencia al documento de transformación http://www.w3.org/TR/2007/NOTE-grddl-primer-20070628/RDFa2RDFXML.xsl, encargado de convertir los fragmentos de RDFa en tal documento a RDF/XML.

Una de las ventajas principales de los modelos de datos RDF es que diferentes modelos distribuidos pueden ser fácilmente combinados en un almacén de RDF. De esa manera, conociendo los lugares de publicación de las agendas de sus amigos, Jane puede combinar y consultar todos los calendarios una vez son transformados en RDF a través de GRDDL. Para ello, utiliza SPARQL que automáticamente combina los datos provenientes de las diferentes fuentes de información de calendarios antes de ejecutar la consulta. El listado 17 muestra la consulta SPARQL que devuelve los detalles (fecha de comienzo, fecha de fin, localización y resumen) de aquellos eventos en los que los amigos Jane, Robin y David coincidirán. El resultado de evaluar esta consulta se muestra en la Tabla 2.

Conclusión

En este artículo hemos revisado tres tecnologías clave que nos ayudarán a acercarnos progresivamente y desde ya mismo a la visión de la Web Semántica: microformatos, RDFa y GRDDL. Los microformatos como hemos visto están diseñados para los humanos en primer lugar y las máquinas en segundo lugar. Ayudan sobremanera a colocar descripciones semánticas en HTML aunque no ofrecen mecanismos estándar de extraer tal información. Su principal problema es que no pueden validarse fácilmente, si mezclamos hCard y hCal no hay modo de garantizar una correcta interpretación, siendo muy específicos a un dominio concreto. RDFa es mucho más adecuado cuando el vocabulario subyacente es demasiado complicado para expresarse en un microformato. Las descripciones en RDFa son mejores para expresar metadatos complejos de forma embebida tales como representación de proteínas o datos geoló-

TABLA 2: Resultado de ejecutar la consulta SPARQL

Fecha Comienzo	Fecha fin	Localización	Resumen
"2007-01-08"	"2007-01-11"	"Edinburgh, UK"	"Web Design Conference"

LISTADO 15 — Calendario de David en XHTML+Embedded RDF

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head profile="http://purl.org/NET/erdf/profile">
    <title>Where Am I</title>
    <link rel="schema.cal" href="http://www.w3.org/2002/12/cal#" />
  </head>
  <body>
    <p class="-cal-Vevent" id="award">
      I then visit Scotland on <span class="cal-dtstart" title="2007-01-08">the 8th
      January</span> to <span class="cal-summary">pick up a lifetime
      achievement award from the Web Design Conference</span>. This time
      the ceremony is in <span class="cal-location">Edinburgh, UK</span>. I'll be taking the train home on the <span class="
      cal-dtend"  title="2007-01-11">10th</span>.
    </p>
    ...
  </body>
</html>
```



XHTML + hCalendar → detect profile → GRDDL source document → fetch transformations → glean-hcal .xsl source and stylesheet → apply transformations → RDF

Figura 6. Conversión de XHTML+hCalendar a RDFx.

LISTADO 16 — Página HTML con RDFa

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
                          "http://www.w3.org/MarkUp/DTD/xhtml+rdfa-1.dtd">
<html xmlns:cal="http://www.w3.org/2002/12/cal/icaltzd#"
xmlns:xs="http://www.w3.org/2001/XMLSchema#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.w3.org/1999/xhtml">
  <head profile="http://www.w3.org/2003/g/data-view">
    <title>Jane's Blog</title>
    <link rel="transformation" href="RDFa2RDFXML.xsl"/>
  </head>
  <body>
    <p about="#event1" class="cal:Vevent">
      <b property="cal:summary">Weekend off in Iona</b>:
        <span property="cal:dtstart" content="2006-10-21" datatype="xs:date">Oct 21st</span> to <span property="cal:dtend"
        content="2006-10-21"  datatype="xs:date">Oct 23rd</span>. See <a rel="cal:url"
        href="http://freetime.example.org/">FreeTime.Example.org</a> for info on <span property="cal:location">Iona, UK</span>.
    </p>
    ...
  </body>
</html>
```

gicos en vez de micro-metadatos más sencillos como eventos o descripciones de personas para los que los microformatos son suficientes. Siempre que se quieran combinar varios vocabularios u ontologías dentro de una página XML o XHTML, RDFa deberá ser nuestra primera opción. Finalmente, GRDDL es una tecnología clave para combinar datos provenientes de microformatos, RDFa o otros formatos semánticos análogos y combinarlos en grafos RDF fácilmente consultables mediantes SPARQL. Actúa como pegamento de los diferentes mecanismos de semantización de información existentes.

En conclusión, la combinación de las tecnologías de microformatos, RDFa y GRDDL nos debería ayudar a progresar hacia la visión de una web con mucho más significado, explotable por las máquinas, no sólo los humanos, sin restringir y dificultar la manera en que creamos contenido web en demasía. Esto es la "web semántica con minúsculas". En la siguiente entrega pondremos los principios de la "web semántica en minúsculas" en la práctica con el desarrollo de un mash-up semántico, mediante el que demostraremos las ventajas no solamente de esta visión más simplista de la Web Semántica sino además el potencial de la combinación de la Web 2.0 con la Web Semántica.

LISTADO 17 — Consulta SPARQL

```
PREFIX ical: <http://www.w3.org/2002/12/cal/icaltzd#>
PREFIX xs: <http://www.w3.org/2001/XMLSchema#>
SELECT ?start1 ?stop1 ?loc1 ?summ1 ?summ2 ?summ3
FROM <http://www.w3.org/TR/grddl-primer/janeschedule.rdf>
FROM <http://www.w3.org/TR/grddl-primer/robin-hcal-grddl.rdf>
FROM <http://www.w3.org/TR/grddl-primer/david-erdf.rdf>
WHERE {
  ?event1 a ical:Vevent;
    ical:summary ?summ1 ;
    ical:dtstart ?start1 ;
    ical:dtend ?stop1 ;
    ical:location ?loc1.
  ?event2 a ical:Vevent;
    ical:summary ?summ2 ;
    ical:dtstart ?start2;
    ical:dtend ?stop2;
    ical:location ?loc2.
  ?event3 a ical:Vevent;
    ical:summary ?summ3 ;
    ical:dtstart ?start3;
    ical:dtend ?stop3;
    ical:location ?loc3.
  FILTER (  ?event1 != ?event2 &&  ?event2 != ?event3  &&  ?event1 != ?event3  ) .
  FILTER ( xs:string(?start1) = xs:string(?start2) ).
  FILTER ( xs:string(?stop1) = xs:string(?stop2) ).
  FILTER ( xs:string(?loc1) = xs:string(?loc2) ).
  FILTER ( xs:string(?start1) = xs:string(?start3) ).
  FILTER ( xs:string(?stop1) = xs:string(?stop3) ).
  FILTER ( xs:string(?loc1) = xs:string(?loc3) ).
  FILTER ( xs:string(?start3) = xs:string(?start2) ).
  FILTER ( xs:string(?stop3) = xs:string(?stop2) ).
  FILTER ( xs:string(?loc3) = xs:string(?loc2) ).
  FILTER ( xs:string(?summ1) <= xs:string(?summ2) ).
  FILTER ( xs:string(?summ2) <= xs:string(?summ3) ).
}
```

Referencias

- Web de Microformats.org – http://microformats.org/
- Introducing RDFa – http://www.xml.com/lpt/a/1691
- GRDDL Primer – http://www.w3.org/TR/2007/NOTE-grddl-primer-20070628/
- Bootstrapping the Semantic Web with GRDDL, Microformats, and RDFa – http://www-sop.inria.fr/acacia/personnel/Fabien.Gandon/tmp/grddl/grddl-introduction-v3/



JavaCup 2008, segunda edición torneo de fútbol virtual Java

ABRAHAM OTERO

El año pasado la revista Sólo Programadores y javaHispano organizamos la primera edición del torneo virtual de fútbol "JavaCup 2007". En el torneo se registraron cerca de 400 participantes y de los equipos enviados 78 fueron aceptados para participar en el torneo. El software del concurso fue descargado más de 3500 veces, siendo el segundo proyecto más activo de javaHispano.net. Se grabaron varias docenas de videos con el resultado de los partidos que, además de en javaHispano.org, fueron publicados en Youtube acumulando varios miles de vistas. Los participantes en el concurso crearon más de 1500 entradas en el foro de la JavaCup durante la competición.

Introducción

El ganador del torneo del año pasado, fue Fabian Nuñez Perez, de Chile con su equipo "F.C. The Patetics"; en segunda posición quedó el madrileño Jorge Cobo López, con "Cobian F.C."; el tercero fue Jacobo Rouces, también español con su equipo "Eclipse" y el cuarto fue el colombiano David Bonilla Bohorquez con "Millos". Las tres nacionalidades de los cuatro ganadores del concurso dan una buena idea del carácter internacional del torneo.

Dados estos resultados, los organizadores consideramos que la JavaCup 2007 fue un éxito y no tuvimos duda en que queríamos repetir la experiencia. Y aquí estamos, un año después, presentando la JavaCup 2008. Algunas cosas han cambiado: el interés que esta competición generó ha hecho que el principal patrocinador del año pasado, Sun Microsystems Ibérica, esta vez haya decidido participar no en calidad de patrocinador sino de organizador. Este movimiento prueba el interés y soporte que la compañía creadora de Java otorga a este torneo (ver Figura 1).

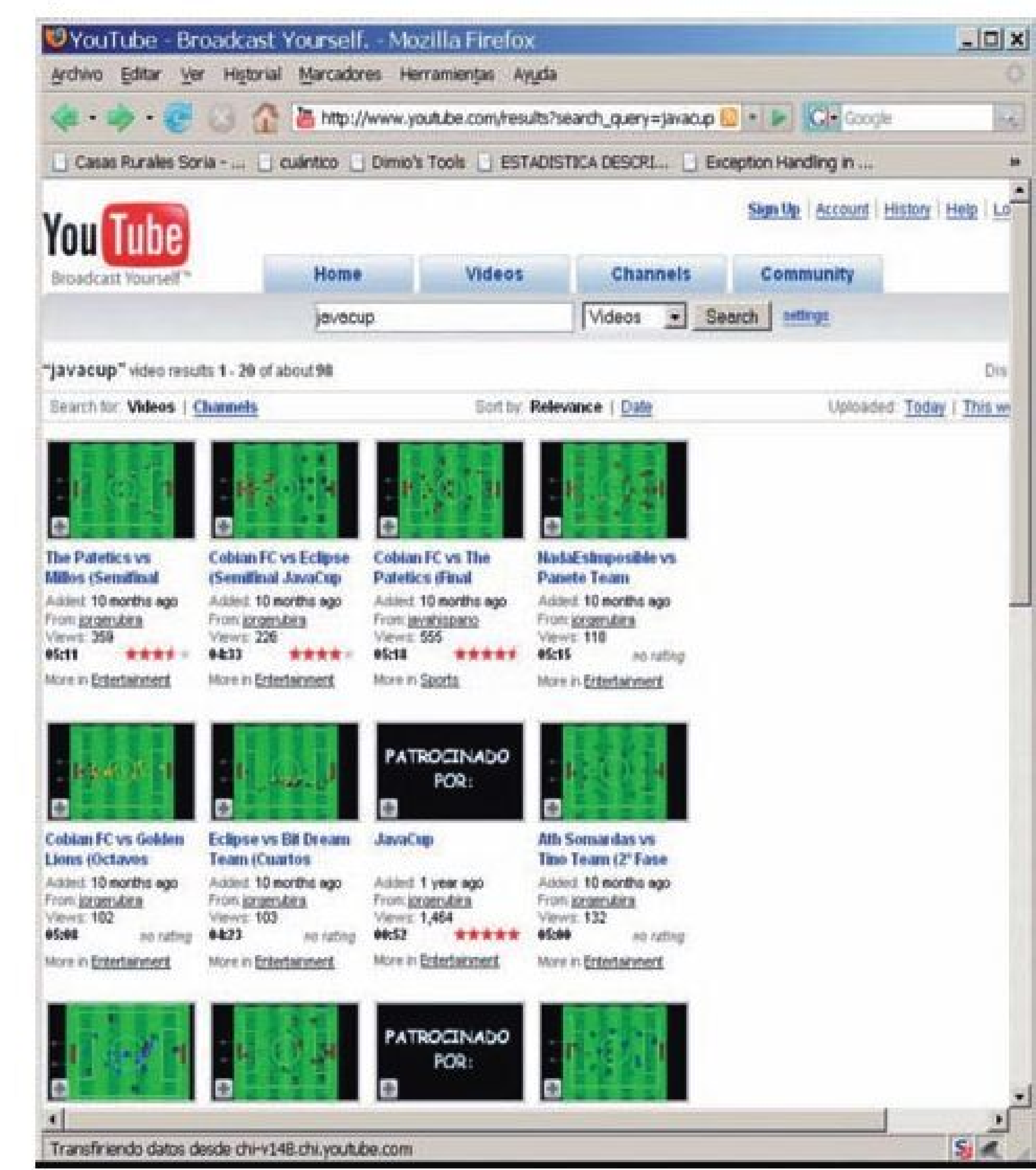
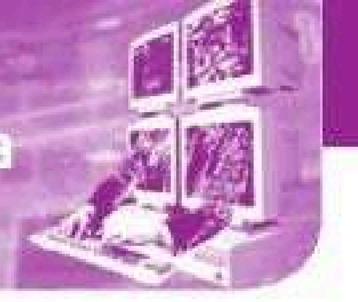


Figura 1: Videos de la JavaCup 2007 en YouTube.

Otro cambio notable es el mayor apoyo empresarial, que a su vez se ha traducido en un mayor número de premios, y premios más sustanciales. Las empresas españolas Kynetia, Everis y Salenda son patrocinadores de oro, plata y bronce, respectivamente, del torneo. La costarricense IsThmus también es patrocinadora de plata. Esto nos ha permitido **triplicar el importe global dedicado a los premios**: el ganador de la JavaCup 2008 recibirá 1500 € y, el segundo 1000, el tercero 500 y el cuarto 250. Además, todos ellos recibirán una suscripción anual gratuita a la versión digital de la revista que estás leyendo ahora mismo.

Si bien ha habido cambios en los organizadores del concurso, en los patrocinadores y en los premios, su espíritu sigue siendo el mismo: un tor-



neo virtual de fútbol donde cada equipo será una clase Java que implementa una interfaz predefinida. Cada participante del concurso debe desarrollar un equipo virtual de fútbol apoyándose en el software que con tal propósito hemos creado. Este software puede considerarse un framework que cuenta con puntos de extensión (que en este caso permiten crear un equipo de fútbol concreto) y ofrece un API en la cual pueden apoyarse las extensiones. El software en el que se basa el concurso se distribuye como un proyecto de Netbeans y como un proyecto de Eclipse; su licencia es GPL y está hospedado en javaHispano.net.

Para crear un equipo de fútbol debe extenderse el framework mediante la construcción de una clase Java que debe implementar una interfaz que define tanto la apariencia de los jugadores como su táctica de juego. Todos los métodos de la interfaz tienen una implementación trivial a excepción del que define la táctica de juego del equipo. Para la implementación de dicha táctica el concursante puede apoyarse en el API que ofrece el software; dicho API permite averiguar la posición del balón, cuál es el jugador de mi equipo que está más cerca del balón, donde está el jugador más cercano a otro jugador dado, etc.

Quizás a un programador novato la idea de construir un equipo de fútbol en Java pudiera parecerle algo complejo y fuera de sus posibilidades. Nada más lejos de la realidad: el entorno de ejecución que ha sido construido para el concurso simplifica notablemente esta tarea.

Como mostraremos en este artículo, 10 minutos son suficientes para la construcción de un equipo que emplee una táctica de juego básica. Por otro lado, la implementación de la táctica de cada equipo puede llegar a ser tan compleja como la pericia e imaginación del concursante le permita. De este modo no se pone una barrera alta en el nivel de conocimientos o dedicación necesarios para participar en el concurso, a la vez que no se limita a potenciales concursantes con conocimientos de la tecnología Java a nivel de gurú y dispuestos a dedicar gran cantidad de tiempo a la implementación de su equipo.

En este artículo mostraremos qué herramientas son necesarias para construir tu propio equipo de fútbol, y dónde conseguirlas. Después, desarrollaremos un equipo de fútbol que emplea una táctica de juego sencilla para que sirva de ejemplo a aquellos lectores interesados en participar en el concurso.

¿Qué necesito para crear mi equipo de fútbol?

Cada equipo de fútbol que participa en la JavaCup es un programa Java. Por tanto, es necesario contar con un kit de desarrollo de aplicaciones Java. Todo el software del concurso es 100% Java, por lo que cualquier compilador y cualquier entorno de desarrollo puede emplearse para participar. Por motivos de comodidad, el software de la JavaCup se distribuye empaquetado como un proyecto de Netbeans y como un proyecto de Eclipse. En el caso de que el participante decida emplear uno de estos entornos de desarrollo, una vez descargado el proyecto y descomprimido podrá abrirlo directamente con el IDE y con sólo hacer clic en el botón de "Play" comenzará a ejecutarse el partido que el software trae cargado por defecto. Si se desea emplear otro entorno de desarrollo, el participante deberá configurar la base de código para ejecutarlo desde él; el código no depende de ninguna librería ni recursos externos por lo que esta tarea es trivial.

En caso de que el concursante sea un programador Java novato y se sienta un poco perdido al leer el párrafo anterior le pondré las cosas más fáciles: le recomiendo que se descargue Netbeans 6.0 o superior; preferentemente la versión que ya viene empaquetada con el JDK de Sun. Puede accederse a esta versión desde la web http://java.sun.com/javase/downloads/index.jsp; la opción de descarga más adecuada sería "App JDK 6 Update X with NetBeans 6.0.1".

Además del entorno de desarrollo Java, es necesario emplear el software que se distribuye bajo licencia GPL y que puede obtenerse en el CD que acompaña a esta revista, en la página web del concurso (http://javacup.javaHispano.org) o en el proyecto JavaCup de javaHispano.net (http://javahispano.net/projects/javacup/).

Por último, hemos construido dos videos explicativos sobre el concurso, videos que pueden encontrarse en el CD de la revista y en la página web del concurso. El primero, de unos 25 minutos de duración, explica qué software se necesita para participar en el concurso, cómo implementar un equipo de fútbol, el API básica que expone el framework y muestra cómo implementar varias tácticas (que se distribuyen junto con el software a modo de ejemplo). Recomendamos a todos los participantes del concurso que vean este vídeo ya que explica cómo construir un equipo con más detalle que este pequeño artículo. El segundo vídeo, de unos 45 minutos de duración, muestra cómo se ha desarrollado el framework que da soporte al concurso. Este vídeo recoge todo el proceso de desarrollo del software, línea por línea. Su contenido no es necesario para aprender a crear equipos y participar en el concurso, aunque puede suponer un excelente material didáctico y satisfacer la curiosidad de aquéllos interesados en saber cómo funciona el software.

Comenzando con el desarrollo de nuestro equipo

Para construir un equipo de fútbol que sea reconocido como tal por el framework de la JavaCup debemos crear una clase Java que implemente la interfaz `futbol.tacticas.concursantes.Tactica` ,cuyo código se puede ver en el listado 1. Esta interfaz permite configurar la apariencia de los jugadores (nombre y colores de la camiseta y pantalones) y definir su táctica de juego. Recomendamos al lector que no se asuste por ver tantos métodos; prácticamente todos se implementan con una línea de código a excepción de aquel que define el comportamiento del equipo. Este será el método donde el lector tendrá que realizar más trabajo y del cual dependerán las posibilidades de que el equipo sea ganador. Para la implementación de este método el concursante puede apoyarse en la API que ofrece el software de la JavaCup; dicha API permite averiguar la posición del balón, cuál es el jugador de mi equipo que está más cerca del balón, dónde está el jugador más cercano a otro jugador dado, etc (ver Listado 1).

El método `public String getNombre()` de la interfaz Tactica debe devolver el nombre del equipo; nuestro equipo se llamará "F. C. Sólo Programadores". El método `public java.awt.Color getColor1()` debe devolver el color de la camiseta del equipo; `public java.awt.Color getColor2()` el color del pantalón; y `public java.awt.Color getColorNumero()` el color del número que se dibujará sobre



LISTADO 1 Interfaz que se debe implementar para construir un equipo

```
public interface Tactica {
    public String getNombre();
    public Color getColor1();
    public Color getColor2();
    public Color getColorNumero();
    public int getFuerza(int n);
    public int getXInicial(int n);
    public int getYInicial(int n);
    public Vector<Comando> getComandos(SituacionJugadores sj);
}
```

cada jugador. Nuestro equipo empleará, por supuesto, los colores de esta revista: azul para el pantalón, rojo para la camiseta y blanco para el número.

A continuación tenemos que implementar un método que devuelva la fuerza con la que es capaz de golpear el balón cada uno de los 11 jugadores del equipo. La declaración del método es: `public int getFuerza(int n)`. Su argumento será el número de uno de los jugadores del equipo (se empieza a contar en 0, por tanto el argumento debe estar comprendido entre 0 y 10) y el dato que devuelve será un valor entre 1 y 10. Debemos tener en cuenta que la velocidad con la que será capaz de correr el jugador será inversamente proporcional a su fuerza; es decir, a más fuerza para golpear el balón menor velocidad para correr. Los participantes deberán intentar buscar el balance más adecuado entre ambos parámetros para que su táctica juegue del mejor modo posible.

También tenemos que implementar dos métodos que devuelven la posición inicial de cada uno de los jugadores del equipo: `public int getXInicial(int n)` y `public int getYInicial(int n)`. Nuevamente, el argumento de estos métodos es el número de uno de los jugadores del equipo y el dato que devuelven es, respectivamente, la coordenada X e Y iniciales de dicho jugador. Nuevamente, cada participante puede jugar con este parámetro para optimizar el rendimiento de su táctica (ver Figura 2).

Para poder trabajar con la posición de los jugadores tenemos que comprender cómo se indican las posiciones en el campo. El campo, para simplificar el desarrollo de los equipos, se ha hecho de dimensión constante. El centro del campo puede considerarse como el origen de coordenadas de un eje imaginario (en la Figura 2 dicho eje de coordenadas imaginario se muestra con líneas punteadas). Los valores positivos sobre el eje X se encuentran hacia la derecha y sobre el eje Y hacia abajo. El campo tiene una longitud de 320 píxeles y una anchura de 240 píxeles. Por tanto, la esquina superior izquierda se corresponde con las coordenadas (-160, - 120); la superior derecha con (-160, 120); la inferior izquierda con (160, - 120); y la inferior derecha, (160, 120). Cuando se ejecuta el programa si movemos el ratón sobre el campo las coordenadas en las cuales se encuentra el ratón se muestran en la esquina inferior izquierda de la ventana; de este modo se simplifica la tarea de decidir dónde queremos colocar nuestro jugadores en el campo (ver Listado 2).

En el código del listado 2 podemos ver una implementación de todos los métodos descritos hasta el momento. Para implementar los métodos que devuelven la fuerza y posición de los jugadores se ha empleado una matriz de 11x3 con nombre `matrizXYFuerza`. Como su nombre indica, la matriz contiene para cada jugador la coordenada X inicial en la posición `matrizXYFuerza [n][0]`, la coordenada Y inicial en la posición `matrizXYFuerza [n][1]` y su fuerza en `matrizXYFuerza [n][2]`

Como el lector habrá podido ver, hasta ahora la implementación de los métodos ha sido muy sencilla. Y sólo nos queda uno por implementar, el más importante, el que define cómo juega cada jugador: `public java.util.Vector<Comando> getComandos(SituacionJugadores sj)`. Este método debe devolver un vector de comandos a ejecutar y recibe como argumento un objeto de tipo `SituacionJugadores`. Este objeto permite conocer la posición de todos los jugadores de tu equipo o del equipo contrario; la posición de la pelota; cuál es el jugador de tu equipo o del equipo contrario que está más cerca de la pelota; cuál es el jugador de tu equipo o del equipo contrario que está más cerca de un determinado punto del campo, conocer si el jugador que está más cerca de la pelota pertenece a tu equipo o al equipo contrario, etcétera. En definitiva, será lo que nos permita obtener cualquier información que podamos necesitar para crear una táctica.

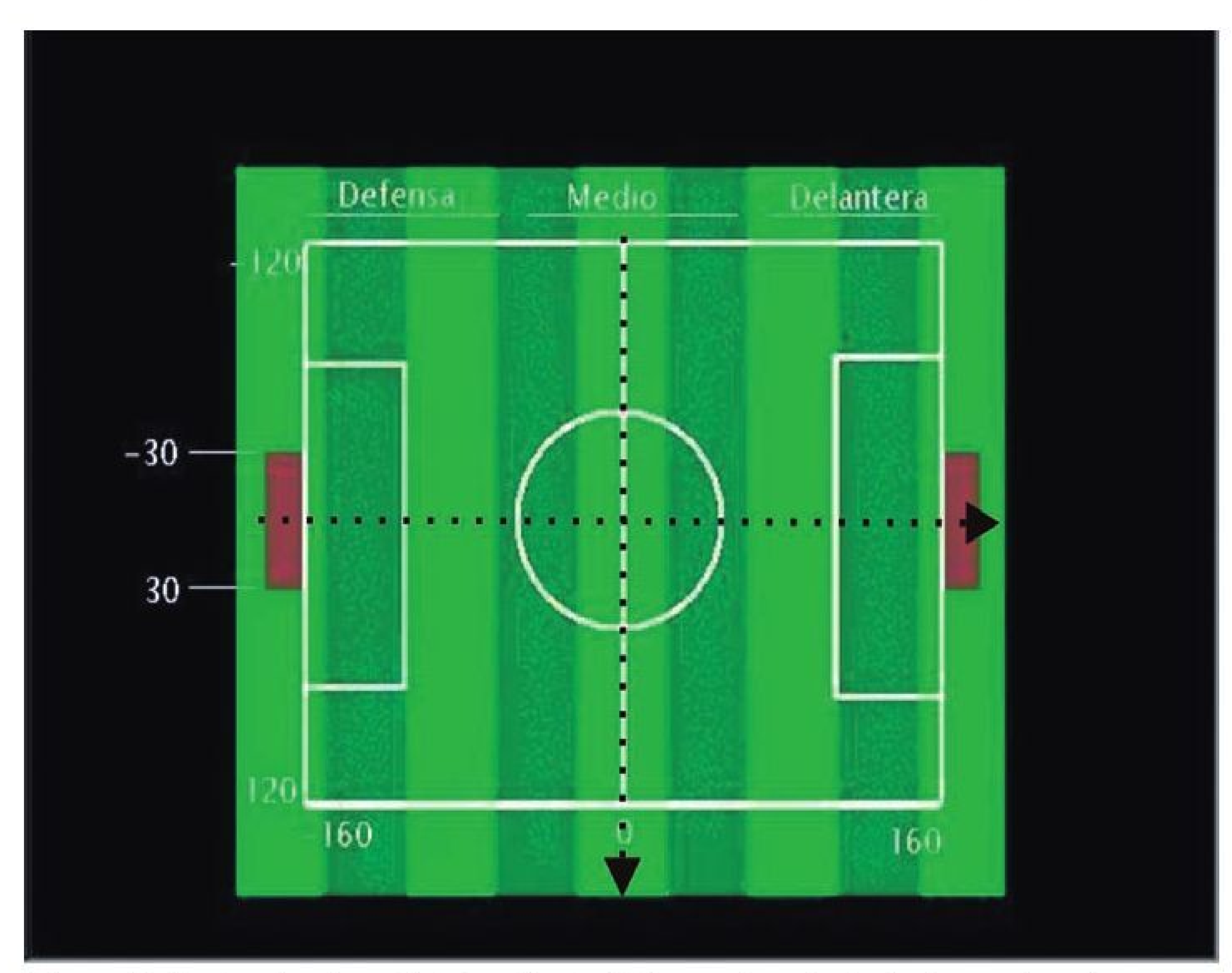


Figura 2: Campo donde se disputan los partidos mostrando el eje de coordenadas que se emplea



LISTADO 2 Todos estos métodos tienen una implementación trivial

```
    private int matrizXYFuerza[][]={
         {-152,0,5},
         {-100,100,7},
         {-120,30,7},
         {-120,-30,7},
         {-100,-100,7},
         {20,100,7},
         {-20,30,7},
         {-20,-30,7},
         {20,-100,7},
         {120,30,7},
         {100,-60,7}  //El nueve se corregirá en el framework y dejará como 7
máximo
     };

     public String getNombre(){
         return "F. C. Sólo Programadores";
     }

     public Color getColor1(){
         return Color.red;
     }

     public Color getColor2(){
         return Color.blue;
     }

     public Color getColorNumero(){
         return Color.white;
     }

     public int getFuerza(int n){
         return matrizXYFuerza[n][2];
     }
     public int getXInicial(int n){
         return matrizXYFuerza[n][0];
     }

     public int getYInicial(int n){
         return matrizXYFuerza[n][1];
     }
```

equipos, elevando la barrera de conocimiento mínimo necesario para poder participar en el concurso (ver Figura 3).

Dentro del método `getComandos (SituacionJugadores sj)` debemos crear los comandos con los parámetros adecuados y añadirlos a un objeto de la clase `java.util.Vector`, que deberá ser el objeto que devuelva el método. El framework de la JavaCup se encargará de ejecutar los comandos, aunque antes comprobará si dicha acción es posible; por ejemplo, entre los comandos podemos indicar que un determinado jugador debe pasarle la bola a otro, pero ese comando sólo se podrá ejecutar si el primero de los jugadores estaba en posesión de la bola.

En el listado 3 podemos ver la táctica de juego empleada por nuestro equipo. Lo primero que se hace es indicarle a cada jugador que vaya a su posición original, que será su posición por defecto a no ser que el jugador reciba otro comando. A continuación, empleando el objeto `SituacionJugadores`, averiguamos cuál de nuestros jugadores está más cerca de la bola y le indicamos a ese jugador que se dirija hacia ella. Después podemos ver los comandos relativos al movimiento del portero. En esta táctica, el portero se mueve en la vertical siguiendo a la bola pero sin abandonar la portería. La portería se sitúa entre las coordenadas -30 y 30; por ello si la posición de

A partir de la información que obtengamos del objeto `SituacionJugadores` debemos decidir qué comandos ejecutamos. Los comandos son instancias de alguna de las clases que derivan de `futbol.tacticas.Comando`. Estas clases son: `ComandoIrA`, que permite indicarle a uno de nuestros jugadores que se desplace a una determinada posición del campo; `ComandoTiroAPuerta`para realizar un tiro a puerta; `ComandoGolpearBola`, para golpear la bola tratando de enviarla hacia una determinada dirección; y `ComandoPase`, para pasar la bola a otro. En la figura 3 podemos ver la jerarquía de clases de comandos. Al usar esos comandos debemos tener en cuenta que si la bola choca contra los límites del campo rebotará de un modo similar a como rebota en un futbolín. Este comportamiento sea definido de este modo para simplificar el desarrollo de las tácticas; el contemplar fueras de juego y córners complica considerablemente la construcción de

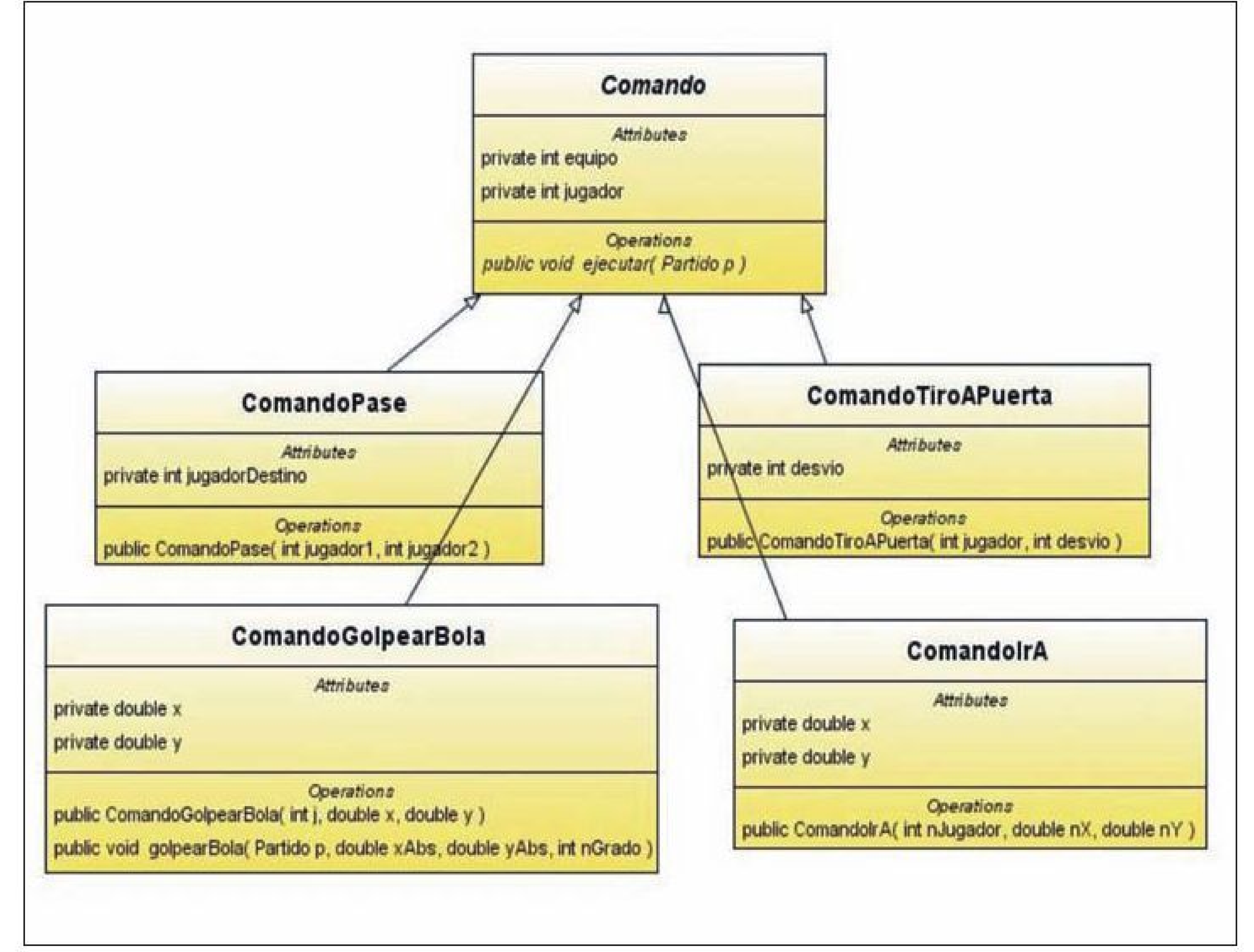


Figura 3: Jerarquía de comandos que se pueden dar a los jugadores.



LISTADO 3 — Éste es el método que definirá la estrategia de juego de nuestro equipo

```
public Vector<Comando> getComandos(SituacionJugadores sj){
        Vector<Comando> comandos=new Vector();
        //Colocamos los jugadores en sus posiciones iniciales
        for (int n=0;n<11;n++)
            comandos.add(new ComandoIrA(n, matrizXYFuerza[n][0],
matrizXYFuerza[n][1]));

        //El jugador más cercano a la bola se dirige hacia ella
        int nJug=sj.getMasCercanoDeBola();
        Point bola=sj.getBola();
        comandos.add(new ComandoIrA(nJug, bola.getX() , bola.getY()));
          //Movimiento del portero
        int y=(int)(bola.getY()/2);
        if (y<-30) y=-30;
        if (y>30) y=30;
        comandos.add(new ComandoIrA(0,matrizXYFuerza[0][0], y));

        for (int n=0;n<11;n++)
            comandos.add(new ComandoTiroAPuerta(n,0));

        return comandos;
    }
```

la bola está fuera de dichas coordenadas el portero se queda junto al "poste" de la portería.

La última parte de nuestra táctica serán los tiros a puerta. En este ejemplo la estrategia que se sigue es muy simple: todos los jugadores tiran a puerta con un ángulo de desvío de 0°; es decir, todos tratan de tirar perpendicularmente a la portería.

El autor se siente en la obligación de advertir a los lectores que el equipo de fútbol que hemos desarrollado pierde escandalosamente al enfrentarse a cualquiera de los cuatro finalistas de la competición del año pasado. Pero bueno, ¡apenas nos ha llevado unos 10 minutos!. Dejaremos como "deberes" para el lector mejorar este equipo.

Probablemente a estas alturas os habéis dado cuenta de que siempre estamos describiendo el juego suponiendo que nuestro equipo es el que juega a la izquierda. ¿Qué pasaría si nuestro equipo jugase a la derecha? No tenéis que preocuparos de ello; el framework se encarga de gestionar esta situación. Siempre que construyamos un equipo debemos suponer que vamos a jugar a la izquierda y será responsabilidad del framework el modificar los movimientos de modo adecuado si nos toca jugar a la derecha.

Como veis, construir una táctica simple es muy sencillo: el framework se encarga de todo el trabajo complicado. Por otro lado, dado que toda la información relativa a la posición de vuestros jugadores, los jugadores contrarios y la pelota está disponible, y dado lo genérico de los comandos que se emplean para jugar podéis construir una táctica todo lo sofisticada que vuestra imaginación y pericia os permita.

Probando nuestro equipo

Para hacer que nuestro equipo de fútbol se enfrente con otro tendremos que modificar el método estático `getTacticas` de la clase `futbol.tacticas.concursantes.Concursantes`. Éste método debe devolver un array que contenga una instancia de los dos equipos que se van enfrentar en el torneo. Por ejemplo, para enfrentar el equipo que hemos creado en este artículo con el equipo `TacticaDemo02`, uno de los equipos de demostración que está incluido dentro del software de la JavaCup, el código del método `getTacticas` necesario para esto se muestra en el listado 4.

Normas básicas del concurso

La participación en el concurso está abierta a todo el mundo y no tiene ningún costo. Para participar basta con registrarse en la web http://javacup.javahispano.org; una



Figura 4 F. C. Sólo Programadores disputando un partido contra Demo02.

LISTADO 4 — Código necesario para crear un partido entre los equipos F. C. Sólo Programadores y Demo02

```
public class Concursantes {

    public static Tactica[] getTacticas(){
        Tactica[] t={new SoloProgramadores(), new TacticaDemo02()};
        return t;
    }
}
```



FIGURA 5: Web de la JavaCup. En ella deberán registrarse los participantes y a través de ella se envían los equipos.

vez el concursantes se ha registrado puede enviar en cualquier momento su equipo para participar en el torneo. Cada participante puede enviar un único equipo. El plazo para el envío de equipos ya se ha abierto y terminará en junio del presente año.

Todos los equipos recibidos se enfrentarán entre sí en un torneo que estará formado por varias etapas eliminatorias; esto es, cada partido que se celebra elimina al concursante que pierde y el vencedor pasa a la siguiente etapa.

El emparejamiento de los equipos se realizará de un modo aleatorio. Si bien el elevado número de envíos que se espera puede impedir grabar en vídeo la ejecución de todos los partidos, al menos los correspondientes a cuartos, semifinal y final se grabarán en vídeo. Los ganadores del concurso se anunciarán en el mes de junio durante el evento OpenJavaDay, coorganizado por Sun Microsystems Ibérica y javaHispano. Dicho evento se celebrará en Madrid en una fecha todavía por determinar. En la web del concurso podéis encontrar las bases del torneo de un modo más detallado y las fechas definitivas de un modo más preciso (ver Figura 5).

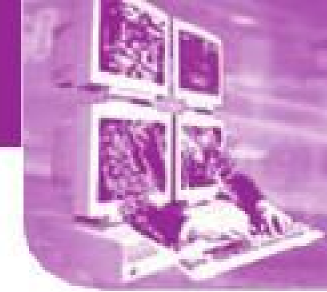
¡Participa!

Participar en la JavaCup es divertido y puede ayudarte a mejorar tus conocimientos de programación Java, además de permitirte ganar premios sustanciales. Para mejorar tu equipo puedes enfrentarlo con equipos de tus compañeros de trabajo o de clases, o con equipos de otros participantes que conozcas a través de los foros de la JavaCup. De ese modo puedes compartir ideas, mejorar tu táctica de juego y conocer gente con intereses similares a los tuyos. Es, en definitiva, una actividad divertida y que fomenta el trabajo en comunidad.

Este concurso es una de las muchas actividades sin ánimo de lucro en las cuales se involucra la organización javaHispano. Nuestro propósito con el concurso es difundir la tecnología Java. Y para ello te pedimos tu ayuda. Si eres estudiante de una universidad imprime el póster-anuncio (podrás encontrarlo en la web del concurso) del concurso y ponlo en alguna de las carteleras de tu Universidad. Si eres un programador, cuélgalo en algún sitio en la oficina. Sólo os estamos pidiendo que gastéis un par de folios y os toméis el tiempo de clavarlos con unas chinchetas en una cartelera, y a nosotros nos ayudará a difundir el concurso.

También recomendamos a los docentes que animen a sus alumnos a descargarse el software y a participar. Les permitirá adquirir destreza en programación de un modo divertido. Crear un equipo puede ser una práctica interesante para una asignatura de programación Java, o para una asignatura de Inteligencia Artificial. Además, todo el software del concurso es software libre. Por tanto tú, docente, y tus alumnos podéis emplearlo y modificarlo libremente.

Muchas gracias por adelantado a todos por vuestra colaboración para difundir esta competición y ¡animaos a crear vuestro propio equipo y enviarlo al torneo! SP



Hibernate y la sencillez de la capa de persistencia en JAVA

DAVID ROLDÁN MARTÍNEZ. Dr. Ingeniero en Telecomunicación, Analista-Programador del ASIC de la Universidad Politécnica de Valencia

En este artículo se explican los conceptos básicos de la construcción de la capa de persistencia de una aplicación multicapa en entorno JAVA. Para ello, se propone una sencilla de aplicación de ejemplo.

La Capa de persistencia en JAVA: Hibernate

En JAVA y en otros muchos lenguajes de programación, resulta bastante común desarrollar aplicaciones orientadas a objetos que trabajen con una base de datos relacional. Se trata de paradigmas bien distintos: mientras que el modelo relacional trabaja con conceptos como relaciones, conjuntos y tuplas, el modelo orientado a objetos se basa en objetos, atributos y asociaciones.

Efectivamente, con JDBC el código necesario para hacer una sencilla consulta que inserte un nuevo registro en una base de datos es una labor tediosa (ver Listado 1). En primer lugar, hay que especificar y cargar el driver JDBC necesario para trabajar con la base de datos concreta y obtener una conexión a la misma; sólo en ese punto ya se está en condiciones de realizar operaciones sobre los datos de las tablas de la base de datos.

Cuando los objetos tienen gran cantidad de atributos, el código JDBC se complica sobremanera. Además, resulta relativamente complicado tratar las asociaciones y manejar las restricciones de clave ajenas. La solución consiste en emplear un mapeador objeto-relacional (ORM, *Object-Relational Mapping*).

Un ORM descarga al programador de todos estos detalles al permitirle trabajar directamente con objetos, de tal forma que el código equivalente al código JDBC anterior, sería el del Listado 2.

Como vemos se trata de un código mucho más sencillo. En este artículo veremos el que es el ORM más maduro y utilizado en JAVA: Hibernate.

Hibernate y Eclipse

Para instalar Hibernate, es necesario descargarse el fichero hibernate-3.2.6.ga.zip y HibernateTools-3.2.0.GA.zip, que contiene las librerías necesarias. Se puede descargar desde la página web de Hibernate (http://hibernate.org). Download-> Binary releases, elegimos Hibernate Core.

El siguiente paso será descomprimirlo en nuestro directorio de trabajo, lo que creará el directorio hibernate-3.0.

Antes de comenzar, vamos a preparar Eclipse para poder utilizar Hibernate, para ello tenemos que Extraer el contenido de HibernateTools-3.2.0.GA.zip en el directorio raíz de eclipse.

Al ejecutar Eclipse podremos ver si se han instalado correctamente los plugins de Hibernate (ver Figura 1).

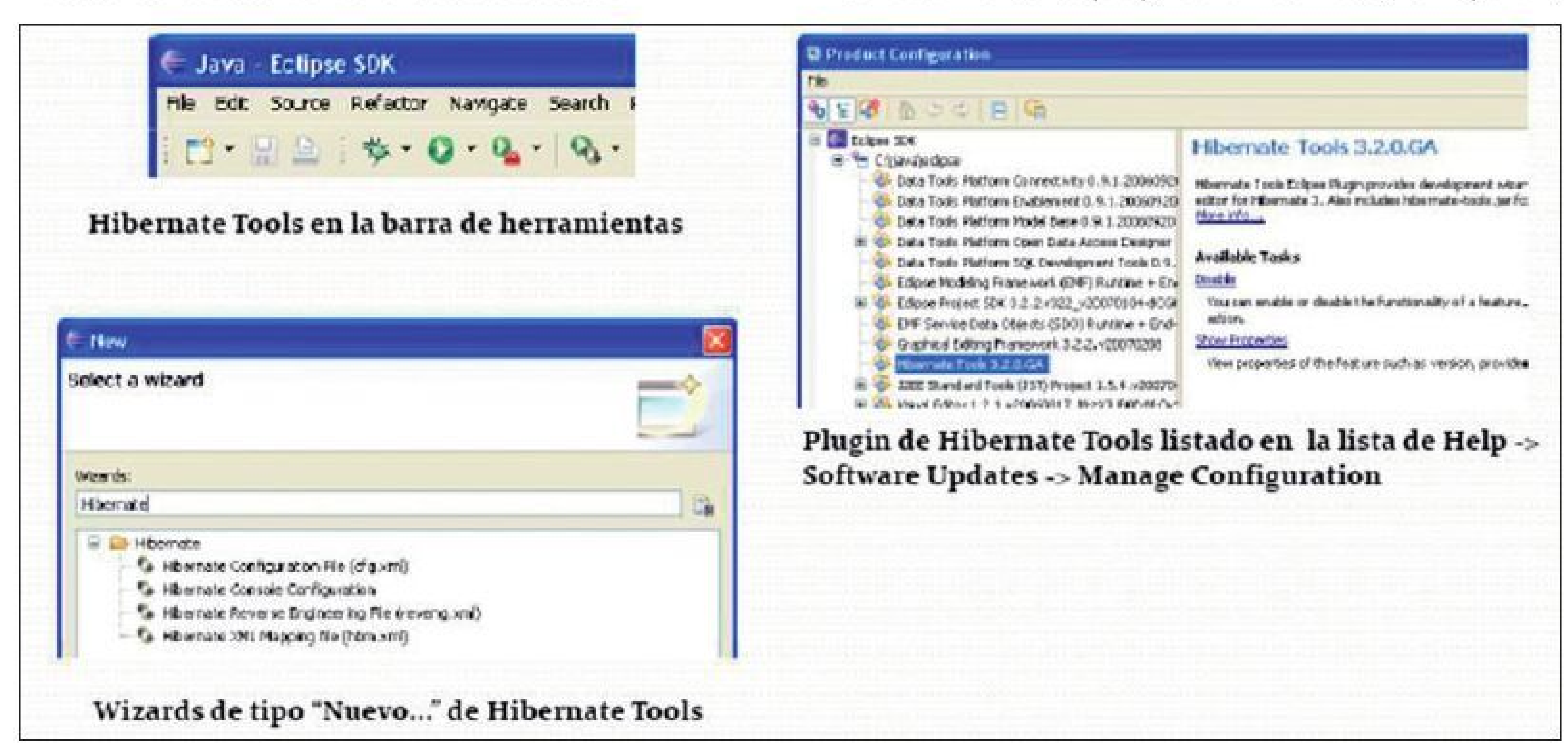


Figura 1. Instalación de las herramientas de Hibernate en Eclipse.



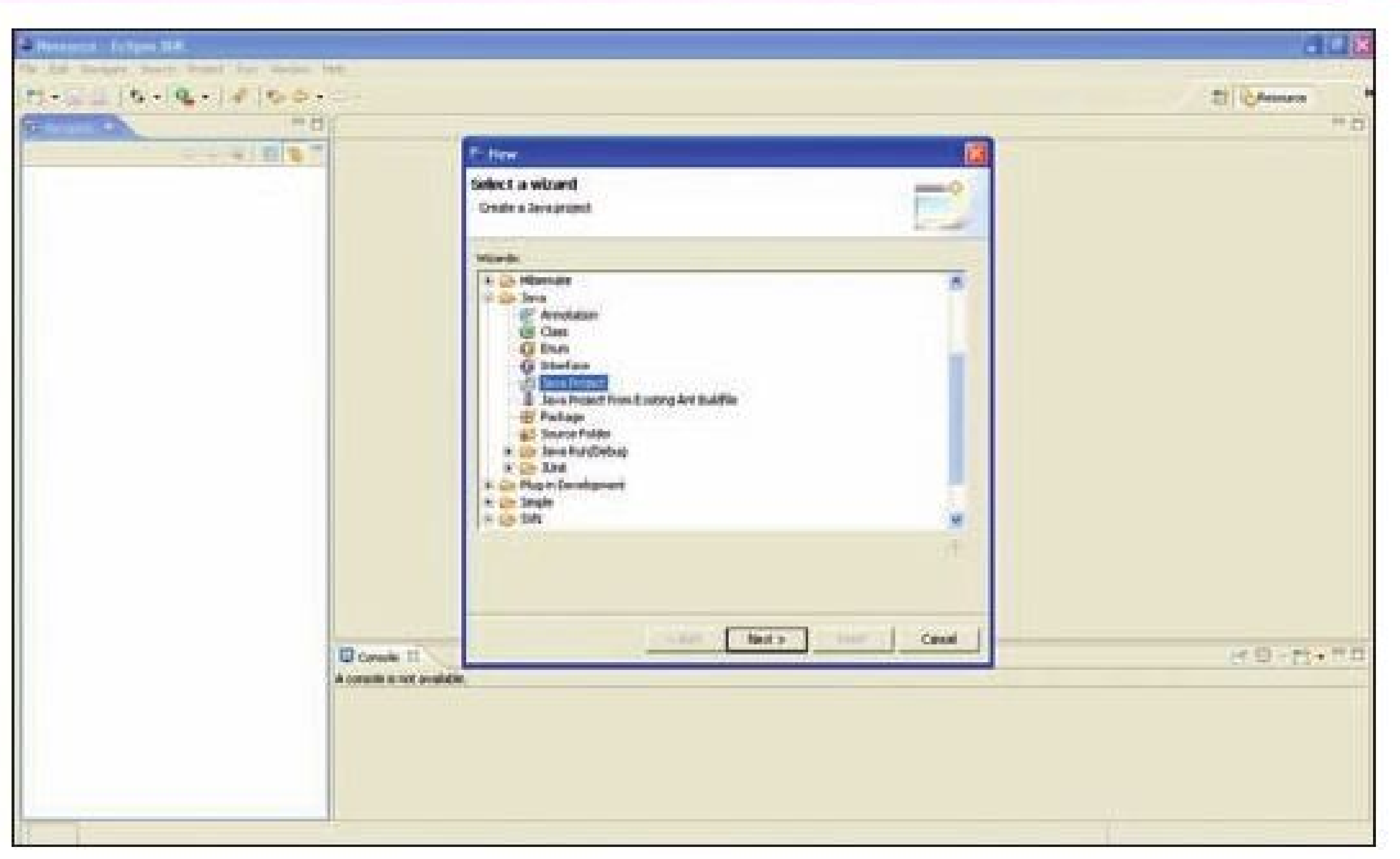


Figura 2. Creación del proyecto JAVA.

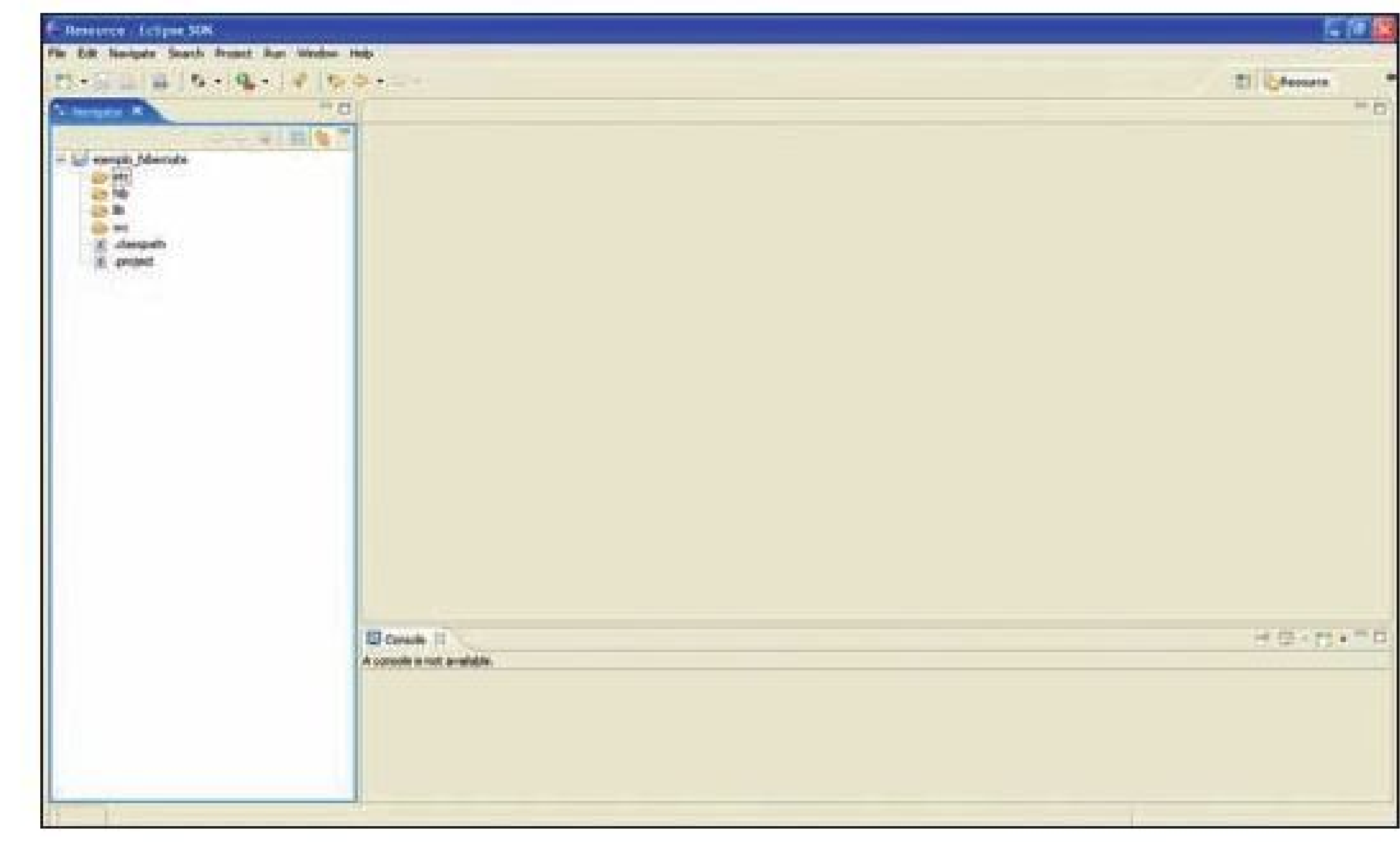


Figura 3. Estructura del proyecto.

LISTADO 1 — Ejemplo de acceso a datos con JDBC

```
try {
Class.forName("org.gjt.mm.mysql.Driver");
}
catch (java.lang.ClassNotFoundException e) {
}

try {
System.out.println("Intentado conectar..");
con = DriverManager.getConnection ("jdbc:mysql//localhost/hibernate", "root", "upa");
System.out.println("Conexión establecida");
String sql = "INSERT INTO Alumnos (dni, nombre, apellidos) VALUE (?,?,?)";
pstmt = con.preparedStatement(sql);
pstmt.setString("111111111");
pstmt.setString("David");
pstmt.setString("Roldán Martínez");
pstmt.executeUpdate();
pstmt.close();
con.close().
}
catch(SQLException ex) {
}
```

LISTADO 2 — Ejemplo de acceso a datos con un ORM (Hibernate)

```
Session session = sf.getCurrentSession();
Transaction tx = session.beginTransaction();
Alumnos al = new Alumnos("111111111", "David", "Roldán Martínez");
session.saveOrUpdate(al);
tx.commit();
session.close();
```

El movimiento se demuestra andando...

Consideramos mucho más didáctico estudiar los entresijos de Hibernate a medida que vamos haciendo nuestro primer ejemplo que describir sus detalles aisladamente.

Lo primero que debemos hacer es crear un proyecto de Eclipse en el que se alojará nuestro código fuente. Para ello, en el menú New->Other...->Java->Java Project y seguimos todos los pasos del asistente (ver Figura 2).

Una vez hecho esto, crearemos la estructura de directorios del proyecto (ver Figura 3).

Seguidamente, añadimos los *.jar externos necesarios en Properties-> Java build path -> Libraries e incluiremos los existentes en hibernate-3.0/lib.

Puesto que la base de datos que utilizaremos en este ejemplo será MySQL, debemos incluir también el mysql-connector-java-5.1.5-bin.jar (puede descargarse desde http://mysql.org) en el Build Path del proyecto, siguiendo unos pasos similares a la

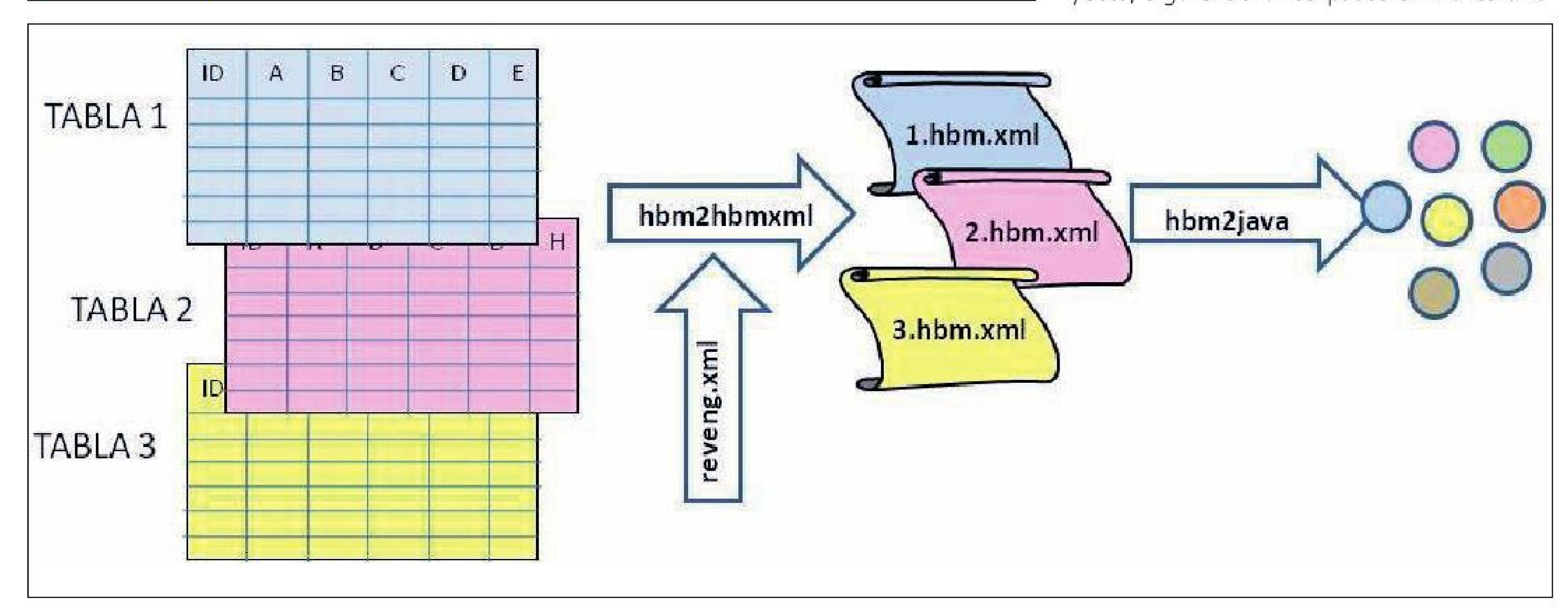


Figura 4. Estrategia bottom-up que seguiremos.



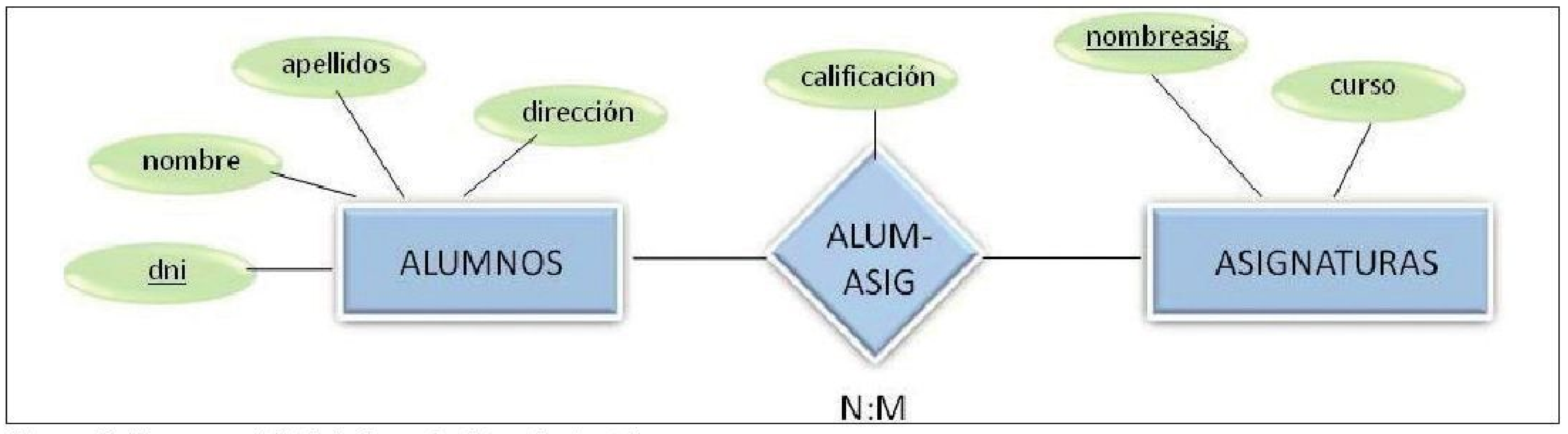


Figura 5. Diagrama E-R de la base de datos de ejemplo.

LISTADO 3 Script de creación de las tablas para una base de datos MySQL

```
CREATE TABLE `hibernate`.`ALUMNOS` (
  `dni` VARCHAR(9) NOT NULL DEFAULT '',
  `nombre` VARCHAR(255) NOT NULL DEFAULT '',
  `apellidos` VARCHAR(255) NOT NULL DEFAULT '',
  `direccion` VARCHAR(255) NOT NULL DEFAULT '',
  PRIMARY KEY(`dni`)
)
ENGINE = InnoDB;

CREATE TABLE `hibernate`.`asignaturas` (
  `nombreasig` VARCHAR(255) NOT NULL DEFAULT '',
  `curso` VARCHAR(255) NOT NULL DEFAULT '',
  PRIMARY KEY(`nombreasig`)
)
ENGINE = InnoDB;
CREATE TABLE `hibernate`.`alum_asig` (
  `dni` VARCHAR(9) NOT NULL DEFAULT '',
  `nombreasig` VARCHAR(255) NOT NULL DEFAULT '',
  `calificacion` FLOAT NOT NULL DEFAULT 0,
  PRIMARY KEY(`dni`, `nombreasig`),
  CONSTRAINT `FK_alum_asig_1` FOREIGN KEY `FK_alum_asig_1` (`dni`)
    REFERENCES `alumnos` (`dni`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `FK_alum_asig_2` FOREIGN KEY `FK_alum_asig_2` (`nombreasig`)
    REFERENCES `asignaturas` (`nombreasig`)
    ON DELETE CASCADE
    ON UPDATE CASCADE
)
ENGINE = InnoDB;
```

inclusión del lib de Hibernate. La instalación de MySQL queda fuera del ámbito de este artículo.

Ya podemos empezar, ¿qué vamos a hacer?

Diseñaremos una base de datos para una Universidad que contenga información sobre los alumnos (DNI, nombre, apellidos y dirección), las asignaturas (nombre) y las carreras (nombre y duración) que se pueden estudiar, teniendo en cuenta las siguientes restricciones:

- Un alumno puede estar matriculado en muchas asignaturas.
- Una asignatura sólo puede pertenecer a una sola carrera, aunque haya asignaturas con el mismo nombre en varias carreras.
- Una carrera puede tener muchas asignaturas.

Para ello, seguiremos una estrategia de tipo *bottom-up*. En una estrategia de este tipo se comienza con un esquema de base de datos y un modelo de datos existentes y, a través de Hibernate, se genera un modelo de objetos que permita desarrollar aplicaciones que hagan uso del modelo de datos en cuestión (ver Figura 4).

Diseño de la base de datos

El modelo Entidad - Relación propuesto se muestra en la Figura 5.

A partir de este modelo, obtenemos el siguiente modelo relacional:

- ALUMNOS (dni, nombre, apellidos, direccion)
- ASIGNATURAS (nombreasig, curso)
- ALUM-ASIG (dni, nombreasig, calificacion)

Por último, sólo es necesario indicar las referencias que se establecen en el modelo en la relación ALUM-ASIG:

- El atributo dni hace referencia a ALUMNOS.dni.
- El atributo nombreasig hace referencia a ASIGNATURAS.nombreasig.

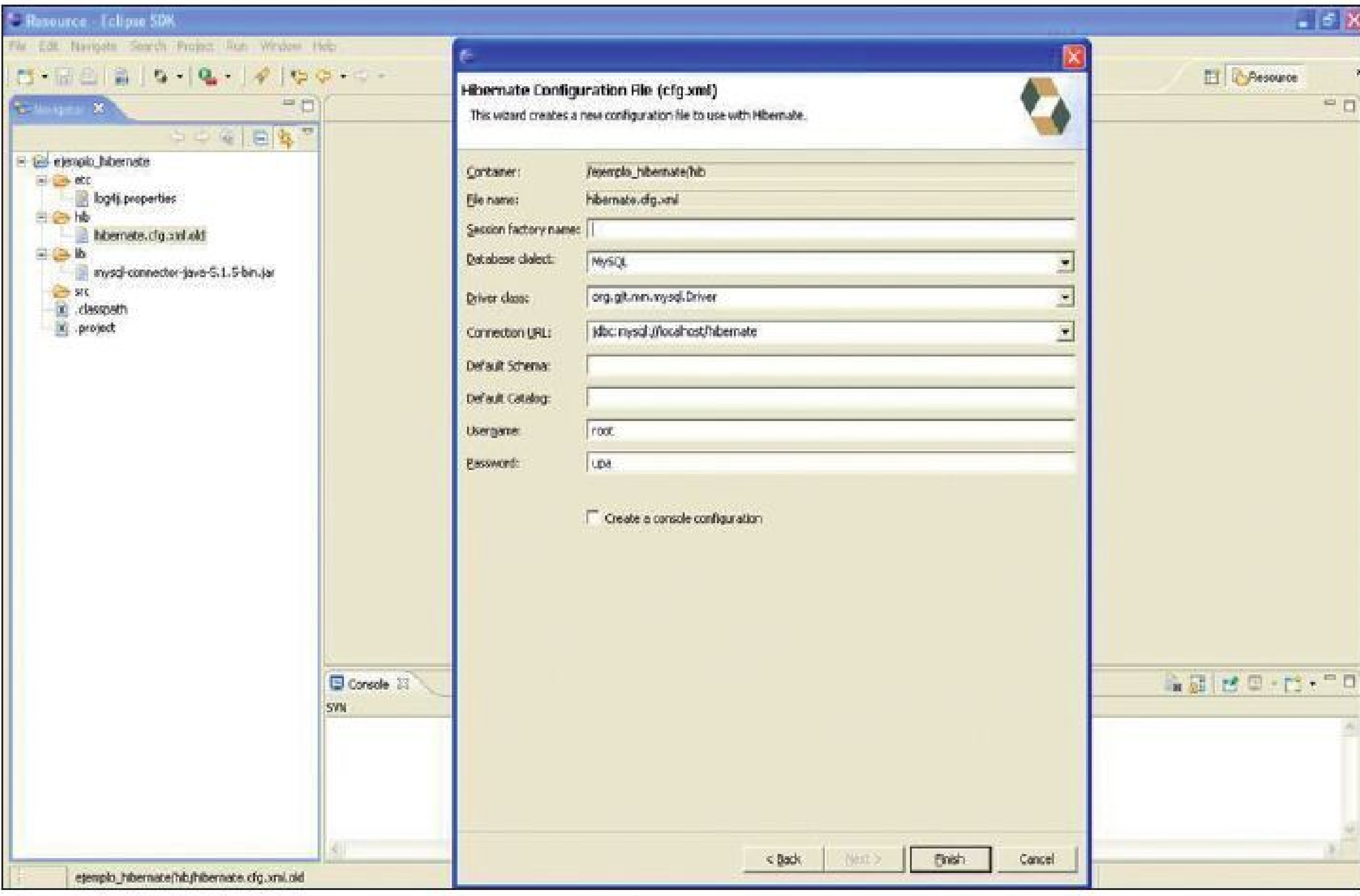


Figura 6. Generación automática del fichero de configuración de Hibernate.



LISTADO 4 — Fichero de configuración de Hibernate generado (hibernate.cfg.xml)

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration
PUBLIC "-//Hibernate/Hibernate Configuration DTD//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
        <session-factory name="prueba">
                <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
                <property name="hibernate.connection.url">jdbc:mysql://localhost/hibernate</property>
                <property name="hibernate.connection.username">root</property>
                <property name="hibernate.connection.password">upa</property>
                <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
                <property name="show_sql">true</property>
        </session-factory>
</hibernate-configuration>
```

El script de creación de esta base de datos es el siguiente (ver Listado 3).

El contenido de este archivo deberá ser ejecutado manualmente sobre la base de datos que se vaya a utilizar utilizando las herramientas propias de dicho manejador. En el caso de MySQL, podemos utilizar la utilidad MySQL Query Browser, la consola o cualquier otra que permita trabajar con la base de datos.

Generación automática de código

Para poder generar código a partir de la base de datos necesitamos crear primero una configuración de consola de Hibernate, y para ello, necesitamos información de la base de datos en forma de un fichero de propiedades o de configuración de hibernate. Para generarlo, podemos utilizar las herramientas de Hibernate para Eclipse (ver Figura 6).

Que da como resultado el fichero del Listado 4, al que hemos añadido la línea <property name="show_sql">true</property> para que muestre información de depuración. En un apartado posterior ampliaremos la información sobre la configuración de Hibernate.

Una vez tengamos el fichero de configuración, creamos la configuración de consola con él utilizando las Hibernate Tools para Eclipse. New->Other...->Hibernate (ver Figura 7):

Y ahora que ya tenemos la configuración de consola, podemos llamar a las tareas de generación de código en Run => Hibernate Code Generation => Open Hibernate Code Generation ... y crearemos una nueva configuración de lanzador (ver Figuras 8 y 9):

Y una vez hayamos seleccionado todo, podremos ejecutar el lanzador y permitir que las herramientas de hibernate nos lo generen todo y actualicen el fichero hibernate.cfg.xml (ver Figura 10).

LISTADO 5 — Código de Alumnos

```
package ejemplo;
import java.util.HashSet;
import java.util.Set;
public class Alumnos implements java.io.Serializable {
        private String dni;
        private String nombre;
        private String apellidos;
        private String direccion;
        private Set alumAsigs = new HashSet(0);
        public Alumnos() {
        }
        public Alumnos(String dni, String nombre, String apellidos, String direccion) {
                this.dni = dni;
                this.nombre = nombre;
                this.apellidos = apellidos;
                this.direccion = direccion;
        }
        public Alumnos(String dni, String nombre, String apellidos,
                        String direccion, Set alumAsigs) {
                this.dni = dni;
                this.nombre = nombre;
                this.apellidos = apellidos;
                this.direccion = direccion;
                this.alumAsigs = alumAsigs;
        }
        public String getDni() {
                return this.dni;
        }
        public void setDni(String dni) {
                this.dni = dni;
        }
        public String getNombre() {
                return this.nombre;
        }
        public void setNombre(String nombre) {
                this.nombre = nombre;
        }
        public String getApellidos() {
                return this.apellidos;
        }
        public void setApellidos(String apellidos) {
                this.apellidos = apellidos;
        }
        public String getDireccion() {
                return this.direccion;
        }
        public void setDireccion(String direccion) {
                this.direccion = direccion;
        }
        public Set getAlumAsigs() {
                return this.alumAsigs;
        }
        public void setAlumAsigs(Set alumAsigs) {
                this.alumAsigs = alumAsigs;
        }
}
```

Aunque las herramientas nos pueden ayudar a generar hasta una página web con la documentación de las clases generadas, nos centraremos en las más importantes: las clases de dominio y utilidad Java y los ficheros de mapping (Hib. XML mappings, Domain y DAO exporters).



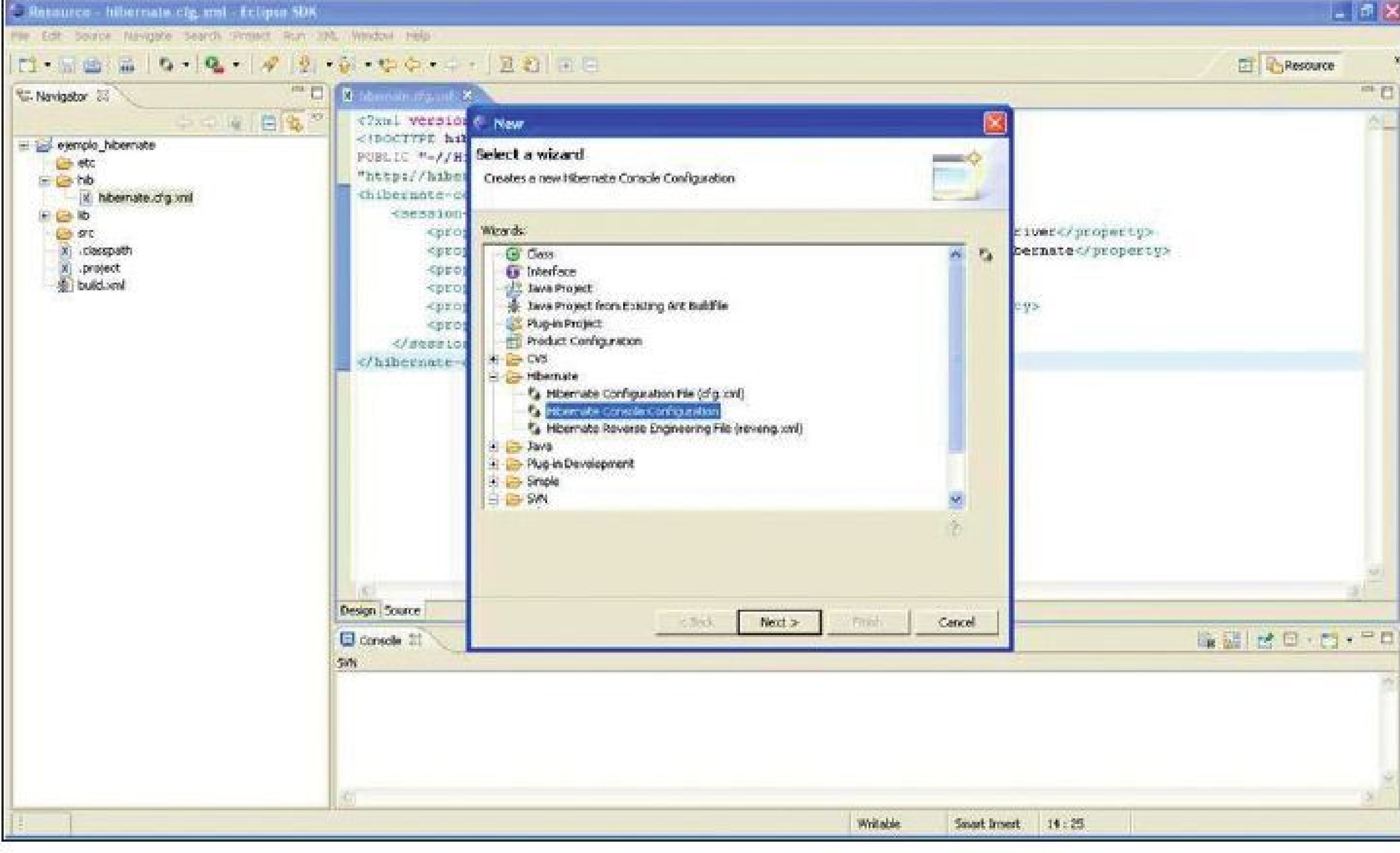


Figura 7. Creación de una configuración de consola.

Las clases de JAVA

Examinemos, por ejemplo, el código de la clase Alumnos.java (ver Listado 5).

Se trata de una clase sencilla que almacena cinco propiedades, cada una de las cuales se refiere a una columna en la tabla alumnos. Además de estas clases, se generan clases de acceso a datos o DAO (Data Access Object). El estudio de estas clases será retomado más adelante.

Los ficheros de mapeo .hbm.xml

Los ficheros de mapeo indican al ORM a qué tabla corresponde un determinado objeto, la asociación entre los atributos del objeto y las columnas de la tabla, así como las relaciones entre el objeto en cuestión y otro objetos, es decir, cómo se modelan las relaciones entre tablas.

Para comprender su importancia y significado, estudiaremos Alumnos.hbm.xml del Listado 6.

Tras la declaración del DTD y la indicación de que se trata de un fichero de mapeo de hibernate a través del elemento <hibernate-mapping>, se indican la clase y la tabla que mapea el fichero en cuestión (elemento <class>). Dentro de éste elemento, se indica la columna de la tabla que constituye la clave primaria (elemento <id>) y, posteriormente, se asocian los atributos del objeto con cada una de las columnas de la tabla (elemento <property>).

Un aspecto importante es el modelado de las relaciones. En este caso, la relación es de tipo <one-to-many>, es decir, que el objeto Alumnos tiene una lista de asignaturas en las que está matriculado.

Clases DAO

Las clases DAO (Data Access Object) se utilizan para instanciar objetos de utilidad añadiendo una sencilla capa de abstracción que encapsule código y facilite el desarrollo. Las herramientas de Hibernate nombran estas clases añadiendo la cadena "Home". Sin embargo, el código DAO debe ser adaptado para cada proyecto, ya que Hibernate genera únicamente métodos genéricos simples. Las operaciones más complejas, deberemos añadirlas nosotros.

Todas las clases DAO, como es el caso de AlumnoHome.java, disponen de una propiedad de tipo org.hibernate.SessionFactory, que permite obtener instancias del objeto Session. Dado su carácter "pesado", suele existir una SessionFactory para toda la aplicación que se crea durante la inicialización de la misma. En definitiva, org.hibernate.SessionFactory es el objeto que permitirá crear sesiones de Hibernate hacia la base de datos.

El modo más sencillo de obtener una SessionFactory es utilizando JNDI, tal y como ha hecho el generador automático de código de Hibernate (ver Listado 7).

Sin embargo, existe otra alternativa y obtenerlo a partir de un objeto Configuration. Una

LISTADO 6 Alumnos.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Generated 22-feb-2008 19:33:22 by Hibernate Tools 3.2.0.CR1 -->
<hibernate-mapping>
    <class name="ejemplo.Alumnos" table="alumnos" catalog="hibernate">
        <id name="dni" type="string">
            <column name="dni" length="9" />
            <generator class="assigned" />
        </id>
        <property name="nombre" type="string">
            <column name="nombre" not-null="true" />
        </property>
        <property name="apellidos" type="string">
            <column name="apellidos" not-null="true" />
        </property>
        <property name="direccion" type="string">
            <column name="direccion" not-null="true" />
        </property>
        <set name="alumAsigs" inverse="true">
            <key>
                <column name="dni" length="9" not-null="true" />
            </key>
            <one-to-many class="ejemplo.AlumAsig" />
        </set>
    </class>
</hibernate-mapping>
```

LISTADO 7 Obtención de una SessionFactory por JNDI

```
private final SessionFactory sessionFactory = getSessionFactory();

protected SessionFactory getSessionFactory() {
        try {
                return (SessionFactory) new InitialContext()
                                .lookup("SessionFactory");
        } catch (Exception e) {
                log.error("Could not locate SessionFactory in JNDI", e);
                throw new IllegalStateException(
                                "Could not locate SessionFactory in JNDI");
        }
}
```



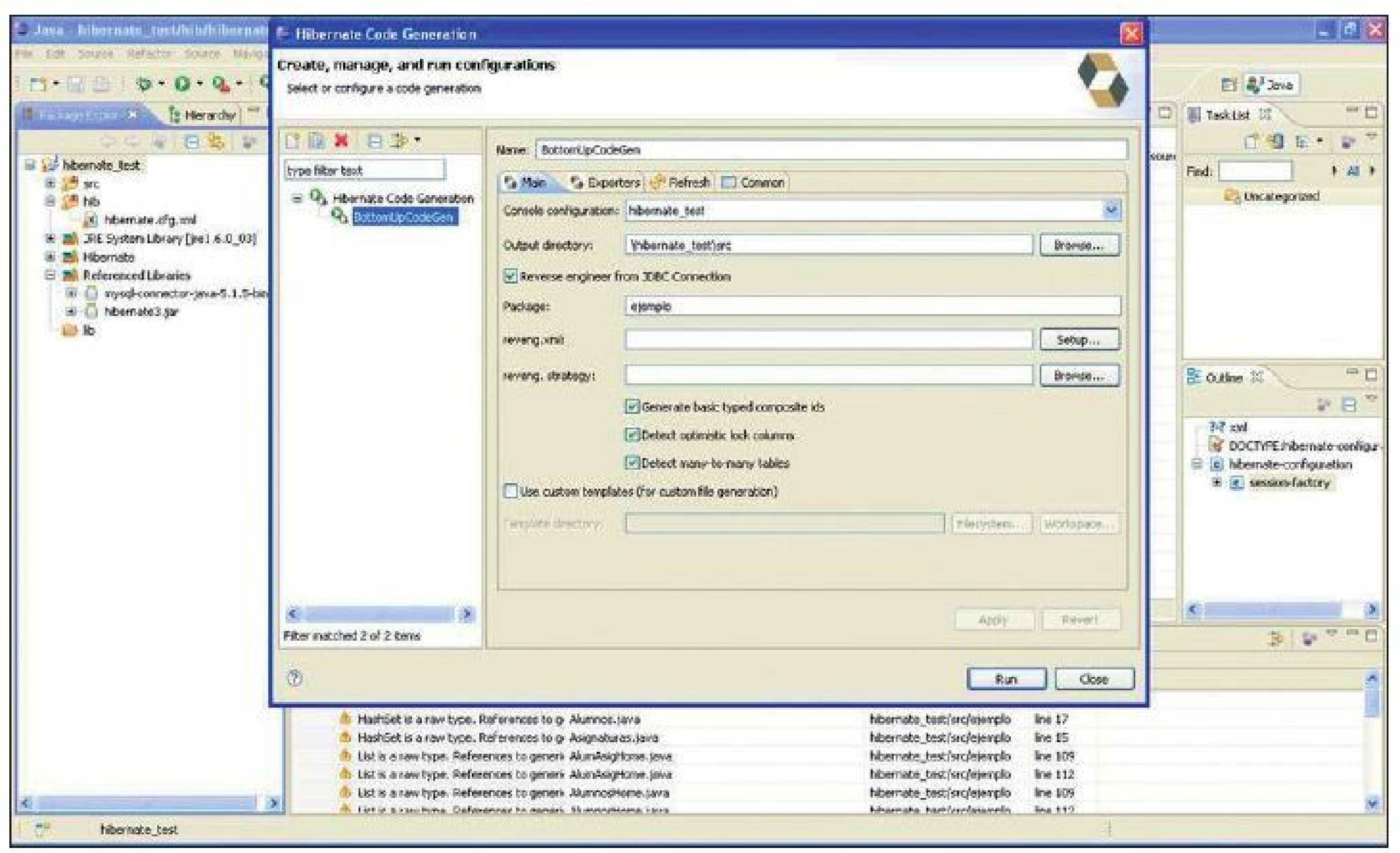


Figura 8. Creación de la configuración del lanzador.

instancia de org.hibernate.cfg.Configuration representa una configuración concreta de Hibernate, es decir, las correspondencias entre el modelo de objeto y el modelo de datos o la ruta de acceso al fichero hibernate.config.xml y a los ficheros de mapeo hbm.xml l(ver Listado 8).

En cualquier caso, el método getSession Factory() simplemente construye la configuración de Hibernate a partir del archivo hibernate.cfg.xml.

Creación de un cliente de prueba

Para estudiar los procedimientos de inserción, borrado y consulta de datos en Hibernate, crearemos un cliente de prueba, llamado TestClient.java.

Inserción y actualización de Datos

El primer paso es incluir en el AlumnosHome crear un método para guardar un objeto de tipo Alumnos. El método save() abre una sesión de Hibernate a partir de la SessionFactory, empieza una transacción sobre dicha sesión, y por último realiza la inserción del dato a través del método saveOrUpdate() de dicha sesión. Como puede verse, se hace uso de un bloque try/finally para asegurar que se cierren la sesión y la transacción (ver Listado 9).

Posteriormente, en la clase TestClient creamos un objeto de tipo Alumnos y lo guardamos utilizando el DAO (ver Listado 10).

Para ejecutar TestClient.java, crearemos una ejecución de Java utilizando el asistente de Eclipse (ver Figura 11).

Si comprobamos la base datos, veremos que, efectivamente, se ha creado un nuevo registro en la tabla Alumnos (ver Figura 12).

LISTADO 8 — Creación de una SessionFactory utilizando la configuración de Hibernate

```
private final SessionFactory sessionFactory = getSessionFactory();

protected SessionFactory getSessionFactory() {
        try {
                return new Configuration()
                        .buildSessionFactory();
        } catch (Exception e) {
                log.error("Could not create SessionFactory ", e);
                throw new IllegalStateException(
                        "Could not locate SessionFactory in JNDI");
        }
}
```

LISTADO 9 — Inserción o actualización de una fila en la tabla Alumnos

```
public void save(Alumnos instance){
        Session session = null;
        Transaction tx = null;
        try{
                session = getSessionFactory().getCurrentSession();
                tx = session.beginTransaction();
                session.saveOrUpdate(instance);
                tx.commit();
        }catch(Exception e){
                tx.rollback();
                log.error("save Alumnos instance failded", e);
        }finally{
                tx.rollback();
                session.close();
        }
}
```

Figura 9. Resultado de la generación automática.

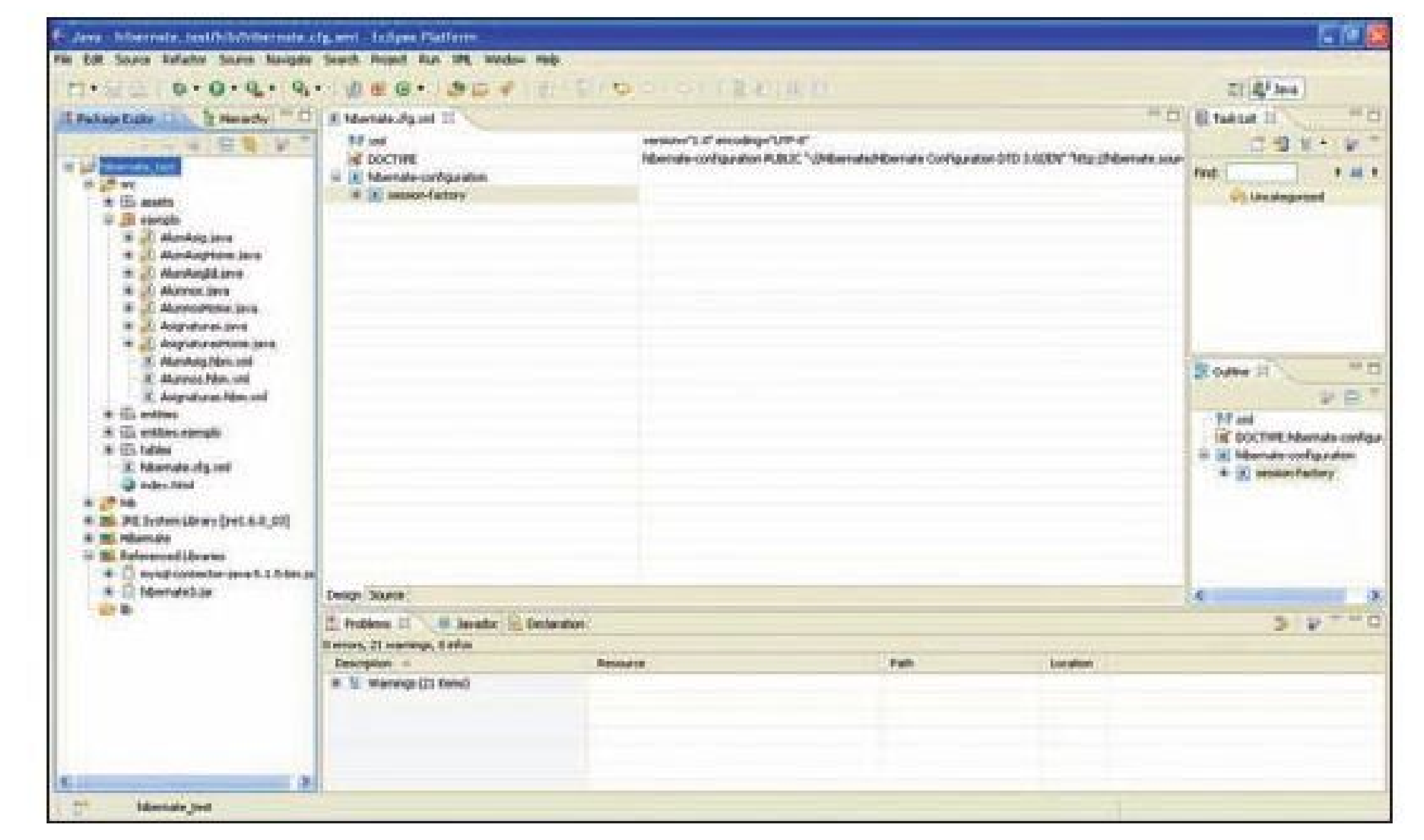


Figura 10. Ejecución del cliente de prueba.



LISTADO 10 — Inserción o actualización de una fila en la tabla Alumnos utilizando el DAO

```
//Obtenemos una instancia del DAO de la tabla alumno
AlumnosHome ah = new AlumnosHome();

//Creamos un objeto alumno
Alumnos alumno = new Alumnos();
alumno.setDni("111111111");
alumno.setNombre("David");
alumno.setApellidos("Roldán Martínez");

//Guardamos el alumno en la bbdd
ah.save(alumno);
```

LISTADO 11 — Consulta de datos

```
//Obtenemos una instancia del DAO de la tabla alumno
AlumnosHome ah = new AlumnosHome();

Alumnos al = ah.findById("111111111");
```

LISTADO 12 — Consulta de datos utilizando el DAO

```
public Alumnos findById(java.lang.String id) {
        log.debug("getting Alumnos instance with id: " + id);
        try {
                Alumnos instance = (Alumnos)
                sessionFactory.getCurrentSession()
                                .get("ejemplo.Alumnos", id);
                if (instance == null) {
                        log.debug("get successful, no instance found");
                } else {
                        log.debug("get successful, instance found");
                }
                return instance;
        } catch (RuntimeException re) {
                log.error("get failed", re);
                throw re;
        }
}
```

Nótese que el identificador generado por la base de datos está disponible automáticamente por medio del método getId() de la clase una vez que la inserción se ha realizado.

Consulta de Datos

El siguiente paso es mostrar cómo se puede hacer la lectura de un registro, por ejemplo, mediante la obtención del registro que acabamos de insertar (ver Listado 11).

El código fuente del método findById() del DAO es el siguiente (ver Listado 12).

El método findById() utiliza una sesión de Hibernate para obtener una instancia de tipo Alumnos a partir de su identificador por medio del método get(),que devuelve null si no encuentra el objeto asociado al identificador solicitado, en lugar de lanzar excepciones como hace el método load(). En este caso, como se trata de una operación de solo lectura, no se utilizan transacciones. Hibernate permite consultar información de la base de datos de muy diferentes maneras. Una de ellas es la utilización de la interfaz org.hibernate.Criteria, que permite construir consultas hacia la base de datos utilizando una perspectiva orientada a objetos.

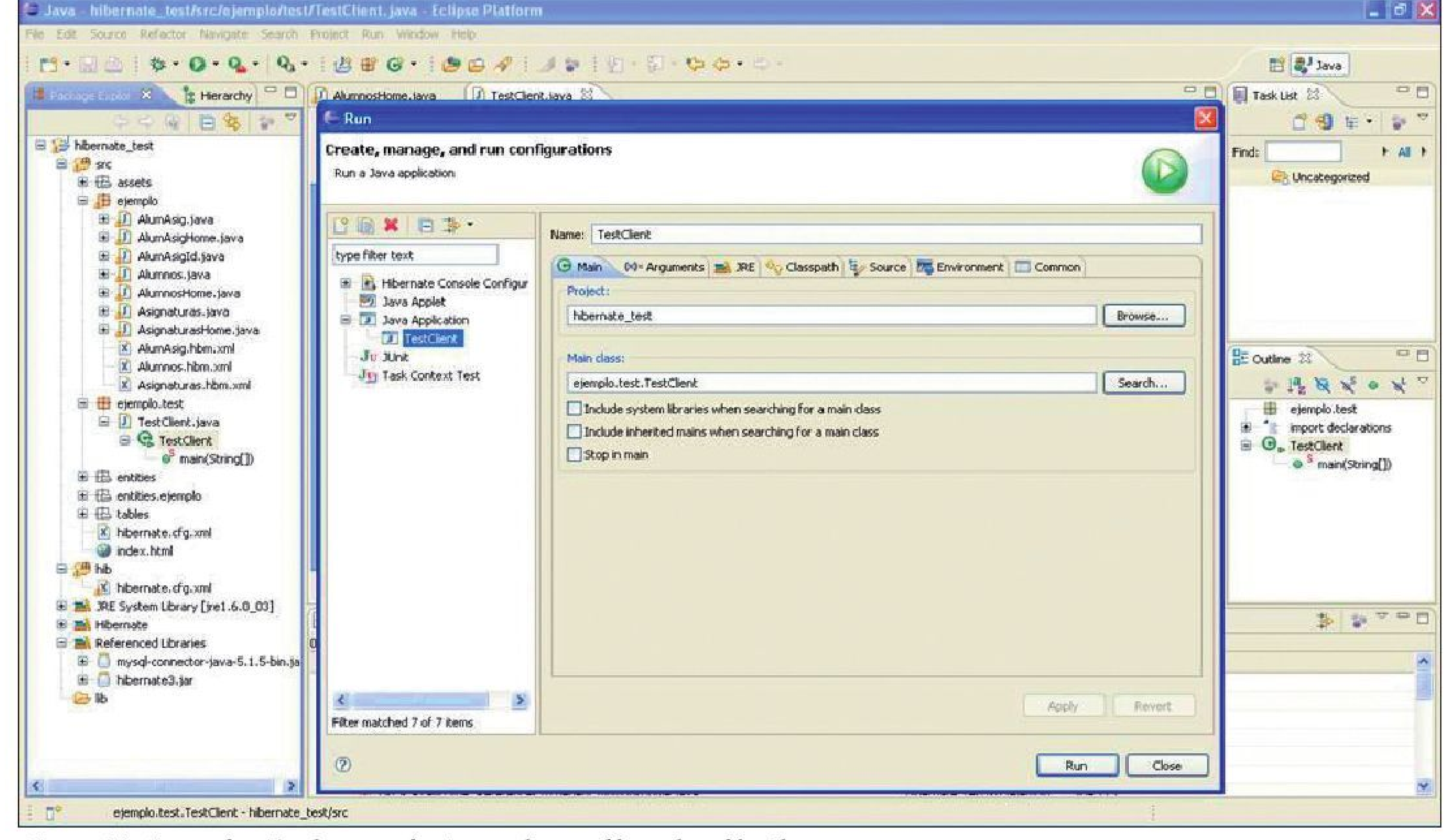


Figura 11. Comprobación de que se ha insertado una fila en la tabla Alumnos.



El método queryAlumno() del DAO muestra un ejemplo de la utilización de esta interfaz. En este caso se construye un objeto Criteria para el tipo de dato Alumnos por medio del método createCriteria() de la interfaz Session. Posteriormente se añade una expresión a la búsqueda que limita los resultados a únicamente aquellos valores cuya propiedad "dni" corresponda al dni pasado como parámetro. El método list() devolverá un listado de todos aquellos objetos que cumplan con los criterios de búsqueda, que será el que devolveremos nosotros. Si se estuviera seguro que obtener un único resultado, se puede utilizar el método uniqueResult() (ver Listado 13).

Otra manera de realizar operaciones sobre los datos es utilizar HQL (Hibernate Query Language). Por ejemplo, el método getAlumnosOrderedList() realiza la consulta de un dato utilizando la interfaz org.hibernate.Query. Esta interfaz permite construir consultas utilizando HQL. En este caso se lee la lista de Alumnos ordenada por el valor del dni (ver Listado 14).

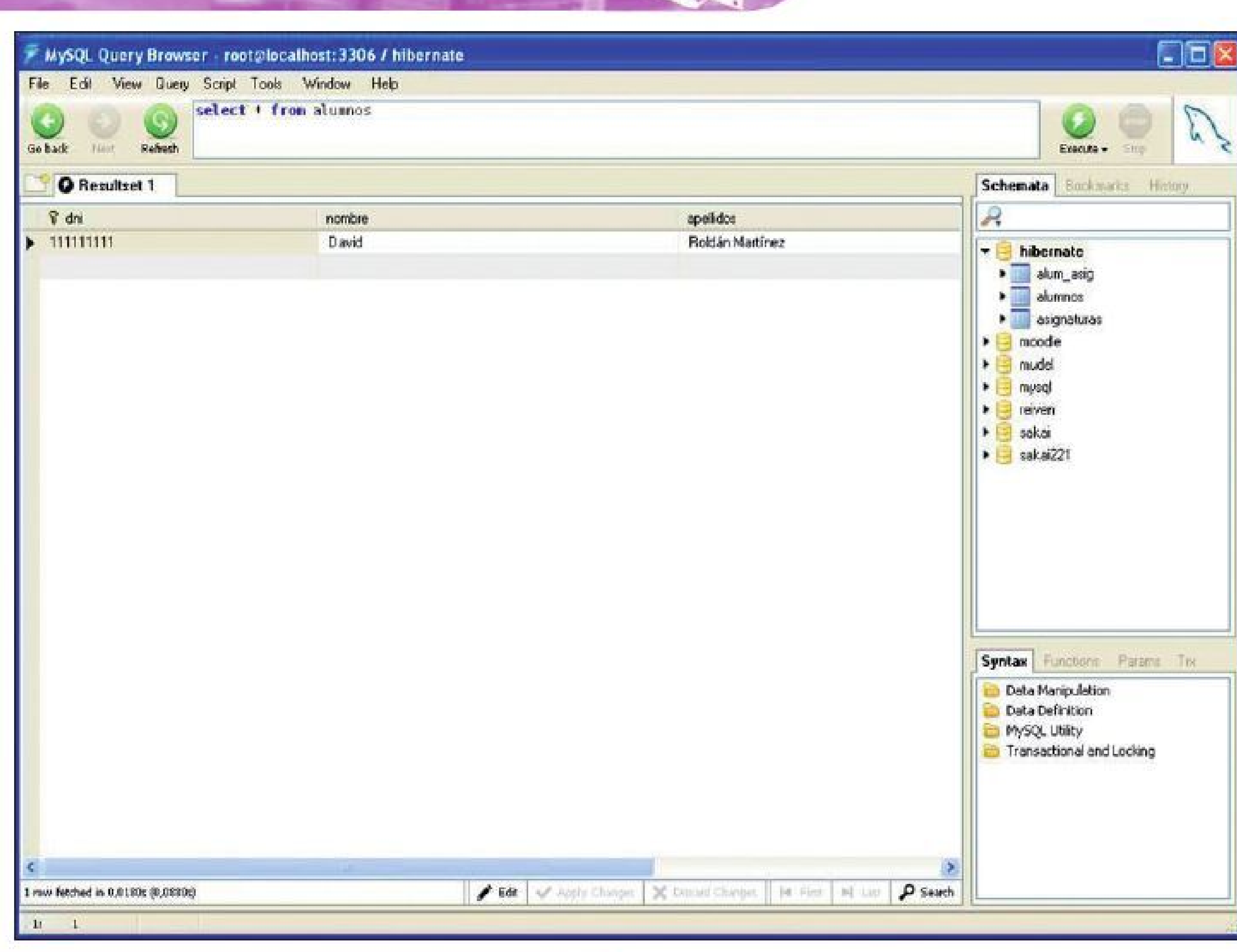


Figura 12.

HQL es un lenguaje muy potente, aunque una descripción detallada del mismo queda fuera del ámbito de este artículo.

LISTADO 13 — Consulta de datos utilizando criterios

```
public Alumnos queryAlumnos(String dni){
        Session session = null;
        List results = null;
        try{
                session = getSessionFactory().getCurrentSession();
                Criteria criteria = session.createCriteria( Alumnos.class );
                criteria.add( Expression.eq( "dni", dni ) );
                results = criteria.list();
                return results;
        }catch(Exception e){
        }finally{
                session.close();
        }
}
```

LISTADO 14 — Consulta de datos utilizando HQL

```
public List getAlumnosOrderedList(){
        Session session = null;
        List results = null;
        try{
                session = getSessionFactory().getCurrentSession();
                Query query = session.createQuery(
                        "SELECT * FROM Alumnos ORDER BY dni");
                results = query.list();
        }catch(Exception e){
        }finally{
                session.close();
        }
        return results;
}
```

LISTADO 15 — Borrado de objetos

```
public void delete(Alumnos persistentInstance) {
        log.debug("deleting Alumnos instance");
        try {
                sessionFactory.getCurrentSession().delete(persistentInstance);
                log.debug("delete successful");
        } catch (RuntimeException re) {
                log.error("delete failed", re);
                throw re;
        }
}
```

Borrado de objetos

Aunque el borrado puede efectuarse utilizando HQL, una manera más sencilla es por medio del método delete() de la interfaz org.hibernate.Session (ver Listado 15).

Conclusión

En aplicaciones complejas que exigen la utilización de bases de datos relacionales y programación orientada a objetos, un mapeador objeto-relacional salva las dificultades derivadas de las diferencias entre ambos paradigmas, simplificando sobremanera las tareas del desarrollador.

En este artículo, se ha explicado el manejo básico del ORM para JAVA por excelencia, Hibernate resolviendo una pequeña aplicación para el manejo de una base de datos de gestión de alumnos y asignaturas. SP





Java Media Framework (y II)

ADOLFO ALADRO GARCÍA, Ingeniero superior de informática.

Con *Java Media Framework (JMF)* resulta muy fácil crear aplicaciones de vídeo y audio. Gracias a los *plug-in* éstas se mantienen actualizadas. Además, uno de los aspectos más interesantes es que no es necesario emplear el reproductor que por defecto ofrece el *API* estándar. Las aplicaciones pueden usar todas las funcionalidades de la clase *Player* y al mismo tiempo crear un diseño totalmente personalizado.

Los formatos y los Plug-in

Uno de los aspectos más interesantes de *Java Media Framework* es el gran número de formatos de audio y video que soporta. En la dirección *java.sun.com/products/java-media/jmf/2.1.1/formats.html* se puede ver la tabla correspondiente a la última versión. Algunos de los más conocidos son: AIFF (*.aiff*), AVI (*.avi*), GSM (*.gsm*), MIDI (*.mid*), MPEG-1 Video (*.mpg*), QuickTime (*.mov*), Wave (*.wav*). Unos de los problemas clásicos de las librerías que permiten la reproducción de audio y vídeo es que rápidamente se quedan obsoletas. Con el tiempo surgen nuevos formatos, o nuevos *codecs* para los formatos ya existentes, de forma que las aplicaciones "envejecen" velozmente y cada vez son menos los ficheros de vídeo o audio que son capaces de reproducir. *JMF* ha resuelto este problema mediante el mecanismo de los *plug-in*. La librería *JMF* se puede actualizar mediante estos pequeños componentes de *software* comúnmente llamados *plug-in*. Así por ejemplo en *java.sun.com/products/java-media/jmf/mp3/download.html* es posible descargar el *plug-in* que permite la reproducción de sonido en formato *MP3*. Una vez que el *plug-in* se instala en el sistema una aplicación *JMF* podría comenzar a reproducir archivos *MP3* sin necesidad de que haya que modificar nada. También es posible el desarrollo de *plug-in* personalizados para algún tipo de formato específico. Sin duda alguna ésta es una tarea bastante más compleja, pero *JMF* abre las puertas a esta posibilidad.

La clase *PluginManager*, perteneciente al paquete *javax.media*, sirve para buscar qué *plug-in* están instalados en el sistema. Asimismo se puede emplear para instalar *plug-in* nuevos. *JMF* distingue entre 5 tipos diferentes de *plug-in* estándar: *DEMULTIPLEXER, MULTIPLEXER, CODEC, EFFECT* y *RENDERER*. Los primeros sirven para extraer pistas concretas de un medio, o para hacer la operación contraria, mezclando varias pistas. Los llamados *codecs* se utilizan para comprimir y descomprimir datos multimedia. Los *plug-in* de tipo *EFFECT* se emplean para alterar de algún modo una pista multimedia (por ejemplo, incluir eco en una pista de audio). Finalmente los *plug-in* del tipo *RENDERER* hacen referencia al dispositivo físico que realmente reproduce el vídeo o el audio. Los métodos estáticos *getSupported InputFormats* y *getSupportedOutputFormats* permiten obtener la lista formatos que un *plug-in* dado admite, respectivamente de lectura y escritura. Recibe dos parámetros. El primero es el nombre de la clase *Java* que se corresponde con el *plug-in*. El segundo es el tipo de *plug-in*. Por último, el método estático *getPluginList*, que puede utilizarse de diversas formas, puede emplearse pasando como *null* los dos primero parámetros y entonces devuelve la lista de todos los *plug-in* de un tipo que hay instalados en el sistema:

```
vectorAllCodecPlugin = getPluginList(null,
null, PluginManager.CODEC);
```

El interfaz *PlugIn*, perteneciente al mismo paquete que la clase *PluginManager*, es el modelo general para todos los *plug-in*:

```
javax.media.PlugIn
   |
   +— javax.media.Codec
   +— javax.media.Demultiplexer
   +— javax.media.Effect
   +— javax.media.Multiplexer
   +— javax.media.Renderer
         |
         +- javax.media.VideoRenderer
```

El método *getName* de la interfaz *PlugIn* devuelve el nombre del *plug-in*.
Por último, los métodos que se emplean para añadir *plug-in* a la instalación de *JMF* son *addPlugIn* y *commit*, que siguen la misma filosofía que los descritos anteriormente. También existe el método *removePlugIn* que se utiliza para eliminar un *plug-in* de la instalación.

Material adicional

El material complementario puede ser descargado desde nuestra web www.revistasprofesionales.com



Personalización del reproductor

Si los *plug-in* permiten que las aplicaciones *JMF* puedan leer todo tipo de formatos de audio y vídeo, la otra característica que hace posible alcanzar un grado notable de personalización es el desarrollo de reproductores a la medida. El componente correspondiente al control del reproductor (el objeto que se obtiene con el método *getVisualComponent* toda vez que se ha producido el evento *RealizeCompleteEvent*) normalmente sólo se emplea con fines didácticos. Lo más frecuente es que las aplicaciones deseen construir sus propios controles, adaptados al diseño de la aplicación, *Web* o de *desktop*. Esos controles personalizados emplean lógicamente los métodos y atributos del objeto *Player*, pero son ellos los responsables de la interfaz. Seguidamente se va a mostrar cómo incorporar unos controles básicos a la aplicación de ejemplo desarrollada en la entrega anterior, *JTestJMF1.java*.

En la aplicación *JTestJMF2.java* la estructura de la interfaz en un poco más complicada. Existen dos paneles (*JPanel*) que se sitúan en la ventana principal utilizando posiciones absolutas:

```
getContentPane().setLayout(null);
...
jPanelVideo.setBounds(0, 0, 640, 400);
...
jPanelControl.setBounds(0, 400, 640, 640);
...
getContentPane().add(jPanelVideo);
getContentPane().add(jPanelControl);
```

Al llamar al método *setLayout* con el parámetro *null* el control para a manos de la aplicación, siendo esta responsable de cómo se colocan los distintos componentes en sus contenedores. El método *setBounds* lleva a cabo precisamente esta tarea. Recibe cuatro parámetros: los dos primeros se corresponden con la posición, y los dos siguientes con la anchura y la altura. Por lo tanto, el primer panel, *jPanelVideo*, se sitúa en la esquina superior izquierda de la ventana principal de la aplicación, *(0,0)*, y tiene 640 píxeles de anchura por 400 píxeles de altura. Este panel será el destinado a mostrar el vídeo. El segundo panel, *jPanelControl*, se corresponde con el área donde van añadirse los controles. Se sitúa en la posición *(0,400)*, es decir, justo debajo del panel anterior.

El *layout* del panel destinado a mostrar el vídeo se hace *null*, ya que en realidad el vídeo se sitúa en la posición *(0,0)* del panel, ocupándolo todo:

```
jPanelVideo = new JPanel();
jPanelVideo.setLayout(null);
```

El *layout* del panel de control es el más simple de todos, *FlowLayout*. Los elementos se disponen en línea y están centrados con respecto al su contenedor gracias al parámetro *FlowLayout.CENTER*:

```
jPanelControl = new JPanel();
jPanelControl.setLayout(new FlowLayout
(FlowLayout.CENTER));
```

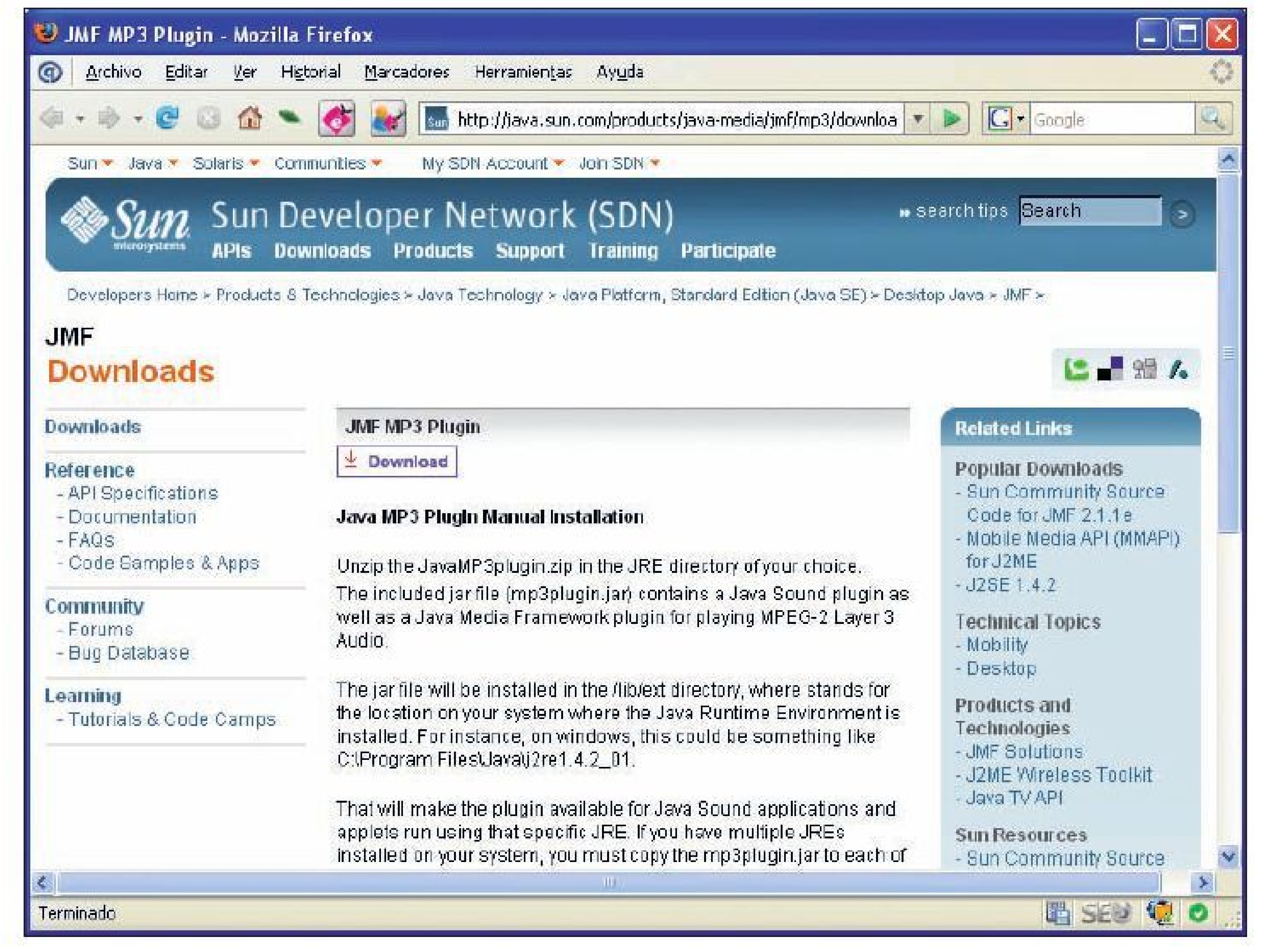
El componente correspondiente al visualizador del vídeo se obtiene y coloca en su sitio cuando se dispara el evento *RealizeCompleteEvent*, que indica que el reproductor ya está listo para empezar a reproducir el vídeo. En esto no existe apenas diferencia con respecto a la primera versión

JMF 2.1.1 - Supported Formats - Mozilla Firefox
Archivo Editar Ver Historial Marcadores Herramientas Ayuda
http://java.sun.com/products/java-media/jmf/2.1.1/formats

Media Type	JMF 2.1.1 Cross Platform Version	JMF 2.1.1 Solaris/Linux Performance Pack	JMF 2.1.1 Windows Performance Pack
AIFF (.aiff)	**read/write**	**read/write**	**read/write**
8-bit mono/stereo linear	D,E	D,E	D,E
16-bit mono/stereo linear	D,E	D,E	D,E
G.711 (U-law)	D,E	D,E	D,E
A-law	D	D	D
IMA4 ADPCM	D,E	D,E	D,E
AVI (.avi)	**read/write**	**read/write**	**read/write**
Audio: 8-bit mono/stereo linear	D,E	D,E	D,E
Audio: 16-bit mono/stereo linear	D,E	D,E	D,E
Audio: DVI ADPCM compressed	D,E	D,E	D,E
Audio: G.711 (U-law)	D,E	D,E	D,E
Audio: A-law	D	D	D
Audio: GSM mono	D,E	D,E	D,E
Audio: ACM**	-	-	D,E
Video: Cinepak	D	D,E	D
Video: MJPEG (422)	D	D,E	D,E
Video: RGB	D,E	D,E	D,E
Video: YUV	D,E	D,E	D,E
Video: VCM**	-	-	D,E
GSM (.gsm)	**read/write**	**read/write**	**read/write**

Terminado

Lista de formatos soportados por la versión actual de Java Media Framework.



Página principal de descarga del plug-in para JMF que permite la reproducción de archivos MP3.



de la aplicación. El objeto *JTestJMF2* comienza a recibir los eventos lanzados desde el objeto *Player* toda vez que se llama al método *addControllerListener*:

```
player.addControllerListener(JTestJMF2.this);
```

Cada vez que se produce un evento se llama al método *controllerUpdate* implementado por *JTestJMF2* y que se define:

```
public synchronized void controllerUpdate
(ControllerEvent controllerEvent)
```

Dentro del método anterior el primer paso consiste en detector qué tipo de evento se ha producido. El *API* distingue entre una gran variedad de tipos. Es muy conveniente conocer todos los eventos que pueden generar el reproductor así como las dependencias jerárquicas que hay entre ellos. El árbol que se muestra en el listado 1 los ilustra poniendo de manifiesto dichas relaciones.

Algunos de estos eventos sólo tienen sentido cuando se emplean *streaming*, es decir, depende del tipo de vídeo o audio. Otros son más comunes y algunos se están viendo en el desarrollo de la aplicación de ejemplo. Así por ejemplo, los eventos *StartEvent* y *StopEvent* marcan respectivamente el comienzo y detenimiento de la reproducción. Volviendo a la aplicación, el evento *RealizeCompleteEvent* en realidad hereda de *TransitionEvent*, que es el evento general que se emplea para modelizar prácticamente cualquier cambio en el estado del reproductor. Con el operador *instanceof* se comprueba el tipo de evento y cuando éste es *RealizeCompleteEvent* se acceder al componente correspondiente al visualizador del vídeo mediante *getVisualComponent* colocándolo en el panel creado para ello.

```
if (controllerEvent instanceof
RealizeCompleteEvent) {
visualComponent=player.getVisualComponent();
  visualComponent.setBounds(0, 0, 640, 400);
  jPanelVideo.add(visualComponent);
}
```

Es importante hacer notar que el panel no tiene *layout* por lo que es preciso utilizar *setBounds* para colocar y dar tamaño al componente.

El Botón Play/Pause

El primer elemento de control que se va a añadir a la aplicación es el típico botón *play/pause*. El vídeo nada más cargarse no comienza a reproducirse, a diferencia de lo que ocurría en la primera versión de la aplicación. La reproducción sólo comienza cuando el usuario hace clic en el botón.

En *Java Swing* el botón clásico es el representado por la clase *JButton*. La instancia de esta clase se crea pasando como parámetro un icono. Posteriormente el botón se añade al panel de control con el conocido método *add*:

```
jButtonPlay = new JButton(icoPause);
...
jPanelControl.add(jButtonPlay);
```

La parte verdaderamente relevante del código es aquella que concierne al procesamiento de los clic del ratón sobre el botón. Los clic se capturan utilizando el método *addActionListener*:

```
jButtonPlay.addActionListener(
  new ActionListener() {
    public final void actionPerformed
    (ActionEvent e) {
      ...
    }
  }
);
```

Cuando el usuario hace clic en este botón de la aplicación el vídeo comienza a reproducirse o se detiene. Esto marca claramente dos estados de la aplicación, y del reproductor en particular. Este estado se guarda en un atributo numérico de la clase *JTestJMF2* denominado *iStatus*. Su valor puede ser *STATUS_PAUSE* (el vídeo está detenido) o *STATUS_PLAY* (el vídeo está reproduciéndose). Si el usuario hace clic en el botón y el vídeo está detenido, el reproductor debe comenzar a mostrar el vídeo, lo que se logra empleando el método *start* de la clase *Player*. Además se modifica el icono del botón, mostrando el icono *icoPlay*, que refleja el nuevo estado del reproductor. Por último se actualiza el valor de la propiedad *iStatus*. Si por el contrario el vídeo está reproduciéndose, se detiene empleando el método *stop*, y además de actualiza el icono del botón y el valor de *iStatus*.

```
if (iStatus == STATUS_PAUSE) {
  player.start();
  jButtonPlay.setIcon(icoPlay);
  iStatus = STATUS_PLAY;
} else if (iStatus == STATUS_PLAY) {
  player.stop();
  jButtonPlay.setIcon(icoPause);
  iStatus = STATUS_PAUSE;
}
```

El último paso que resta para completar la funcionalidad del botón consiste en

LISTADO 1

```
javax.media.ControllerEvent
  |
  +- AudioDeviceUnavailableEvent
  +- CachingControlEvent
  +- ControllerClosedEvent
  |     |
  |     +- ControllerErrorEvent
  |     |     |
  |     |     +- ConnectionErrorEvent
  |     |     +- InternalErrorEvent
  |     |     +- ResourceUnavailableEvent
  |     +- DataLostErrorEvent
  +- DurationUpdateEvent
  +- FormatChangeEvent
  |     |
  |     +- SizeChangeEvent
  +- MediaTimeSetEvent
  +- RateChangeEvent
  +- StopTimeChangeEvent
  +- TransitionEvent
        |
        +- ConfigureCompleteEvent
        +- PrefetchCompleteEvent
        +- RealizeCompleteEvent
        +- StartEvent
        +- StopEvent
              |
              +- DataStarvedEvent
              +- DeallocateEvent
              +- EndOfMediaEvent
              +- RestartingEvent
              +- StopAtTimeEvent
              +- StopByRequestEvent
```



actualiza el valor de *iStatus* y el icono del botón cuando el vídeo llega a su fin, simultáneamente "rebobinando" el vídeo hacia atrás para que pueda volver a comenzar la reproducción. El evento lanzado por el objeto *Player* que marca el final de la reproducción se denomina *EndOfMediaEvent*:

```
} else if (controllerEvent instanceof
EndOfMediaEvent) {
  player.setMediaTime(new Time(0));
  jButtonPlay.setIcon(icoPause);
  iStatus = STATUS_PAUSE;
}
```

Con la llamada a *setMediaTime* con el parámetro *new Time(0)* el vídeo vuelve a su posición inicial.

El tiempo transcurrido

Otro de los controles típicos que puede tener todo reproductor es aquel que muestra tanto la duración total del vídeo si este está parado como el tiempo transcurrido de reproducción, dinámicamente mientras ésta tiene lugar. La duración total del vídeo se obtiene empleando el método *getDuration* de la clase *Player*:

```
Time timeDuration = player.getDuration();
```

El valor devuelto es un objeto de la clase *Time* perteneciente al paquete *javax.media*. Esta clase permite registrar el tiempo y su precisión se mide en nanosegundos. Con el método *getSeconds* se accede al valor en segundos. Se emplea un método estático, *getDurationNumberFormatter*, para convertir ese valor en una cadena apta para ser mostrada como reloj del vídeo:

```
private final static DecimalFormat
getDurationNumberFormatter()
{
  DecimalFormatSymbols decimalFormat
  Symbols = new DecimalFormatSymbols();
  decimalFormatSymbols.setDecimal
  Separator(':');
  DecimalFormat numberFormatter = new
  DecimalFormat("00.00");
  numberFormatter.setDecimalFormat
  Symbols(decimalFormatSymbols);
  return numberFormatter;
}
```

La clase *DecimalFormat*, del *API* estándar de *Java*, permite dar formato a los números de una forma rápida y cómoda. La cadena que el constructor recibe como parámetro simboliza dicho formato. Los ceros ('0') representan números. Si por ejemplo el valor de un número fuera pequeño y no tuviera dos dígitos a la izquierda de la coma, al utilizar un cero se indica que se pinte cero, ya que no existe dígito. El punto ('.') es la coma decimal. En el caso del reproductor interesa cambiar este símbolo por los dos puntos (':') y para ello es necesario emplear la clase *DecimalFormatSymbols*, que permite personalizar el objeto *DecimalFormat*.

El campo que muestra el marcador de tiempo es de tipo *JTextField*. El valor que recibe como parámetro hace referencia al tamaño que dicho componente debe adoptar en forma de número de caracteres. Con el método *setFont*, y partiendo del tipo de letra que por defecto tiene asignado el componente, se establece un tipo de letra algo más grande de lo normal.

```
jTextFieldDuration = new JTextField(4);
jTextFieldDuration.setFont(jTextField
Duration.getFont().deriveFont(25f).
deriveFont(Font.BOLD));
...
jPanelControl.add(jTextFieldDuration);
```

El ultimo paso, como siempre, consiste en añadir el componente al panel correspondiente.

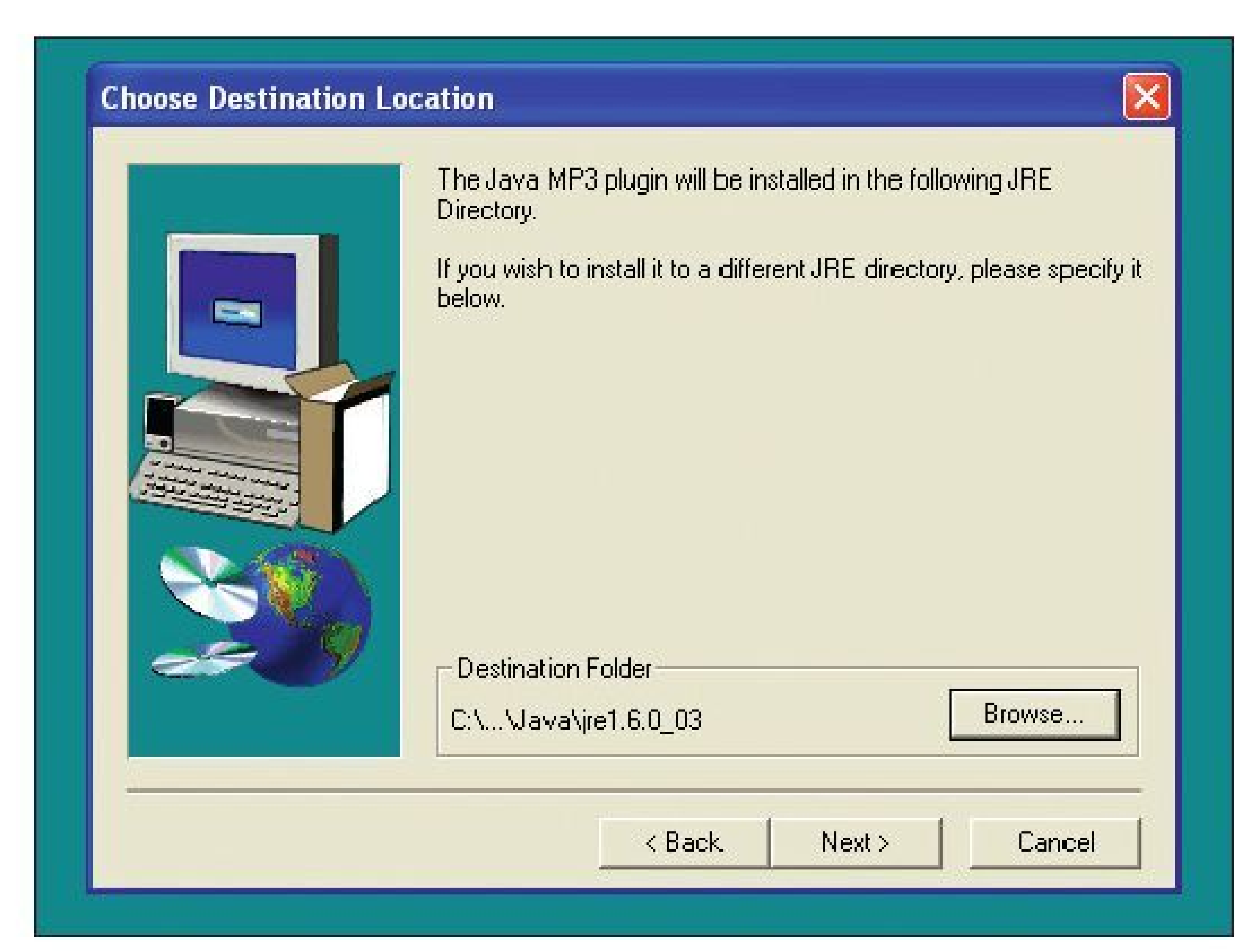
Actualiazación automática con un *Timer*

Lógicamente el tiempo transcurrido es un valor que se necesita actualizar constantemente junto con la reproducción del vídeo. Una de las maneras más simples de hacerlo consiste en la utilización de un *Timer* del *API* estándar de *Java* que periódicamente consulte al *Player*. La clase *Timer*, perteneciente al paquete *javax.swing*, además garantiza el buen rendimiento de la aplicación en el sentido de que no bloquea el *thread* de ejecución principal.

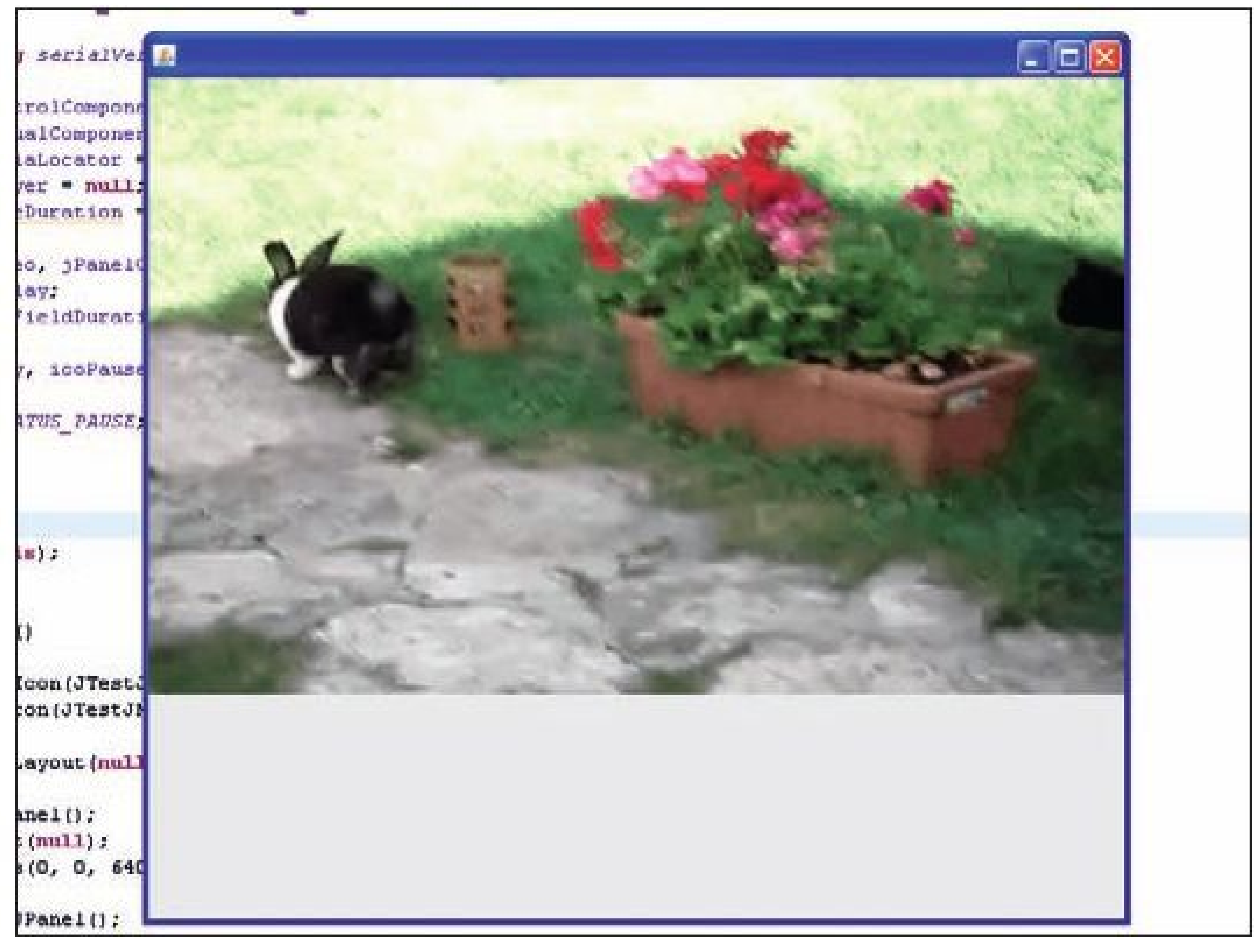
De nuevo volviendo al método *controllerUpdate*, la aplicación captura el evento *StartEvent*, que es el que marca el inicio de la reproducción del vídeo y por lo tanto el momento a partir del cual hay que actualizar el valor del campo de texto que muestra el tiempo transcurrido:

```
} else if (controllerEvent instanceof
StartEvent) {
  if (timer==null) {
    timer = new Timer(1000, ···);
    timer.start();
  }
}
```

En ese momento se crea un nuevo objeto de la clase *Timer* y posteriormente se arran-

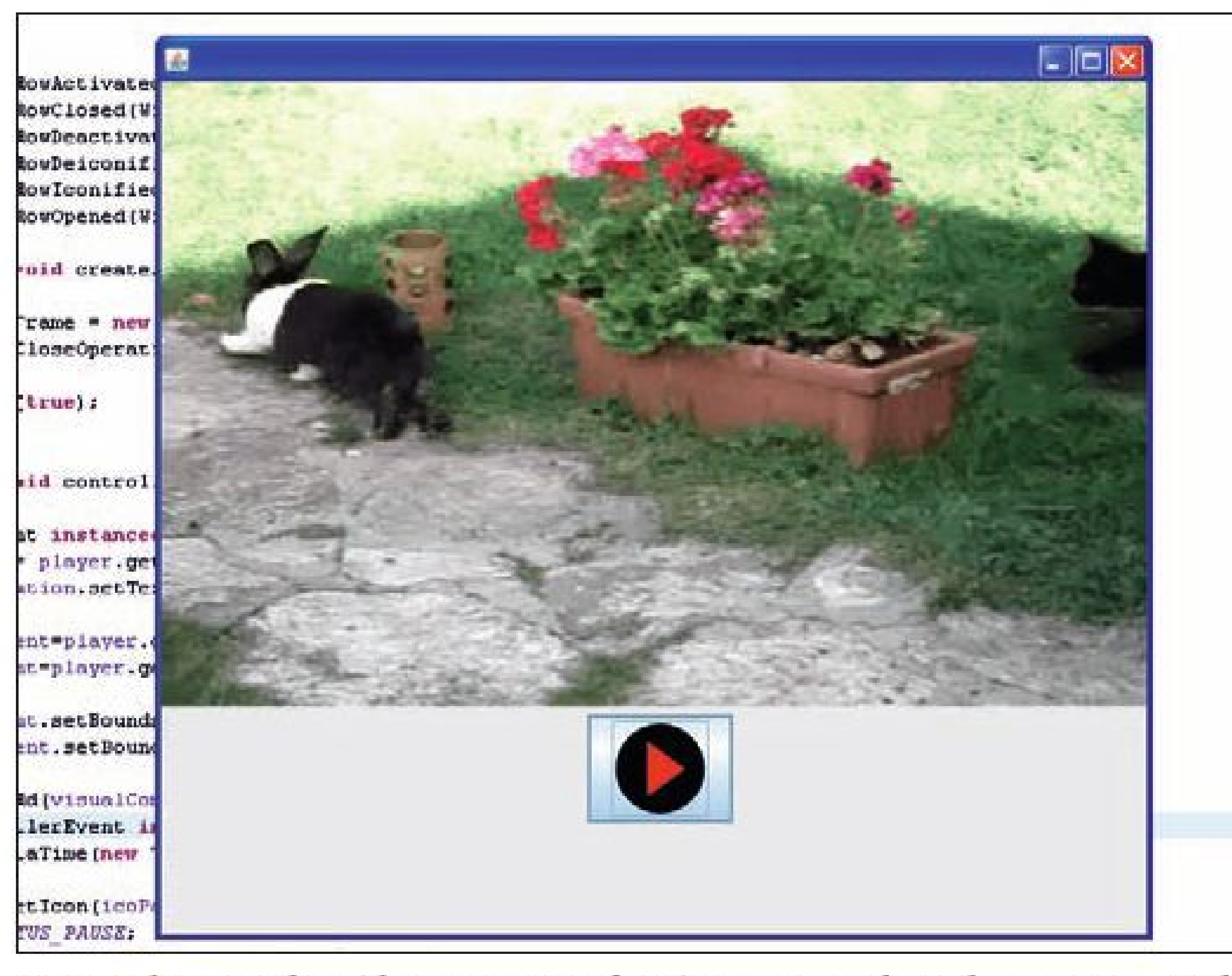


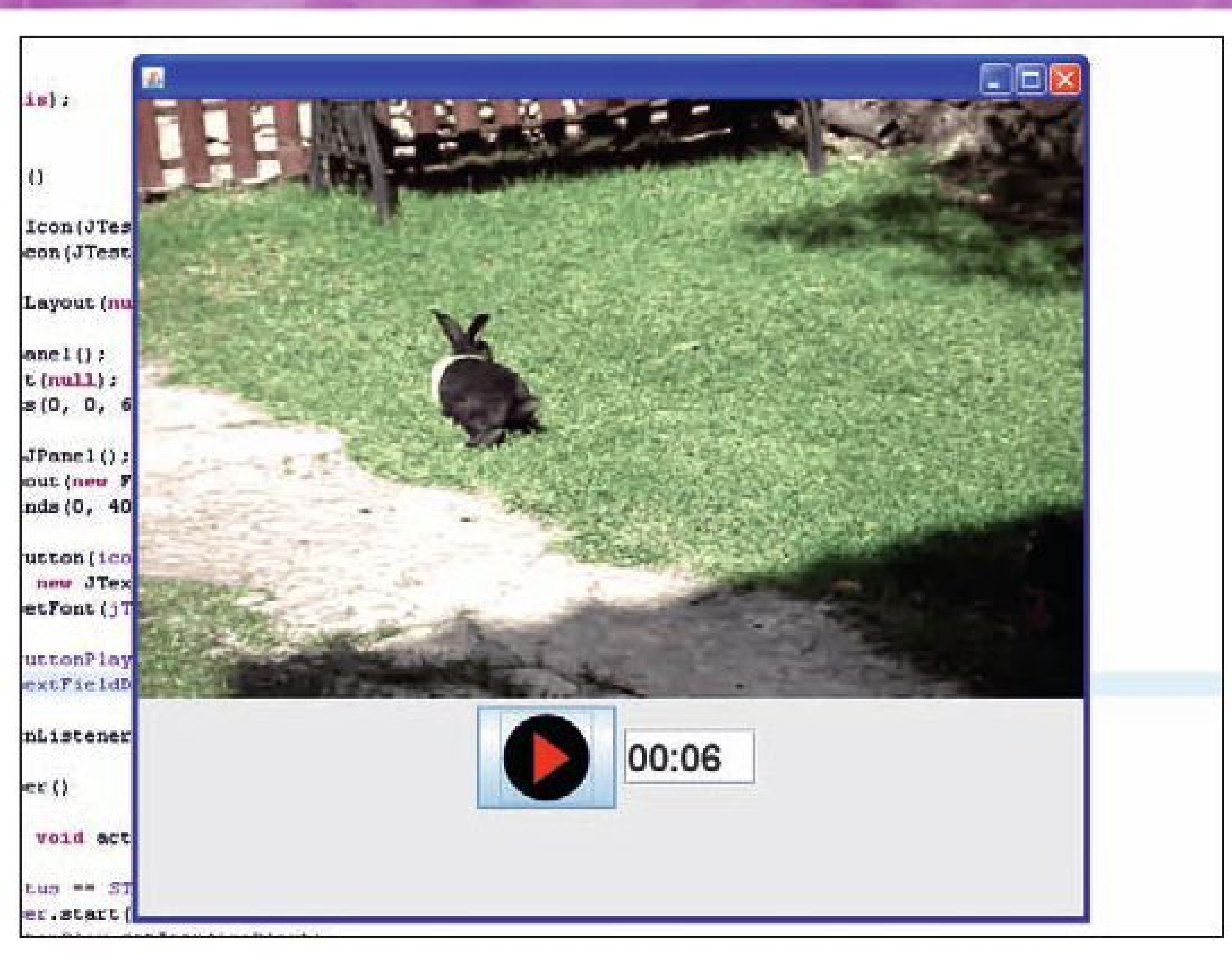
Instalación del plug-in para JMF en Windows.



Reproductor de vídeo sin el componente de control que ofrece por defecto.




Reproductor de vídeos con un botón personalizado que controla la reproducción.



Reproductor de vídeos con un marcador que muestra el tiempo transcurrido.

ca el temporizador empleando para ello el método *start* de la clase *Timer*. El primer parámetro del constructor es la frecuencia, es decir, el tiempo en milisegundos que transcurre entre llamadas del temporizador; o dicho de otra forma, ese temporizador lanzará una llamada cada segundo (1000 milisegundos). El segundo parámetro es un objeto de una clase que implementa la interfaz *ActionListener*:

```
new ActionListener() {
  public void actionPerformed
  (ActionEvent actionEvent) {
    ...
  }
}
```

El temporizador ejecutará el método *actionPerfomed* cada segundo y es ahí, en el cuerpo de ese método, donde debe actualizarse el valor del campo de texto.

```
String sDuration = durationNumberFormatter
(player.getMediaTime().getSeconds());
jTextFieldDuration.setText(sDuration);
```

El tiempo transcurrido del vídeo se obtiene mediante el método *getMediaTime*, y con el método *getSeconds* se obtiene el valor en segundos. Con método creado para dar formato a lo segundos se logra la cadena de texto con la que luego se actualizar el campo de texto, usando el método *setText*. Los temporizadores creados con la clase *Timer* se ejecutan fuera del *thread* principal de ejecución de la aplicación, pero la ejecución del método *actionPerformed* sí ocurre en ese *thread* principal. Por ello es seguro utilizar ahí el método *setText*, o cualquier otro típico de las aplicación con *Java Swing*.

Cuando el usuario hace clic en el botón para detener la reproducción del vídeo, hay que detener también el temporizador. Esto puede hacerse o bien en la gestión del clic del botón o en la gestión del evento correspondiente en el reproductor:

```
if (timer!=null) {
  timer.stop();
  timer = null;
}
```

A decir verdad no es totalmente necesario detener el temporizador ya que aunque el valor del campo de texto se siga actualizando, como el vídeo no avanza, se actualiza con el mismo valor siempre. Esto es transparente al usuario, ahora bien, si realmente no es necesario que el temporizador se ejecute, conviene detenerlo para no consumir recursos de la máquina de una manera innecesaria.

Terminación "limpia" de las aplicaciones

Es realmente importante incidir en el hecho de que las aplicaciones deben programarse de forma que ocurra lo que ocurra durante su ejecución al terminar se liberan los recursos. Si lo anterior es siempre primordial, lo es doblemente en aplicaciones de audio y vídeo, que además usan temporizadores y otros recursos varios, en general pesados.

En la aplicación JTestJMF2 la terminación viene marcada por la ejecución del método *windowClosing*, que es el que se dispara cuando el usuario cierra la ventana principal de la aplicación. El primer paso consiste en detener el temporizador:

```
try {
  if (timer!=null) timer.stop();
  timer = null;
} catch (Exception e) {
}
```

El siguiente paso consiste en detener el reproductor:

```
try {
  if (player!=null) player.stop();
} catch (Exception e) {
}
```

Y finalmente se llama al método *deallocate* garantizando así que *JMF* libera todos los recursos implicados en la creación del reproductor y la reproducción del vídeo:

```
try {
  if (player!=null) player.deallocate();
} catch (Exception e) {
}
```

Conclusión

A lo largo de estos dos artículos se ha hecho una introducción muy somera a la tecnología *Java Media Framework*, especialmente en lo que tiene que ver con la reproducción de audio y vídeo. En realidad detrás de las siglas *JMF* hay mucho más: grabación y reproducción utilizando todo tipo de dispositivos, conversión entre formatos, etc. Parece claro que el futuro inmediato pasa necesariamente por las aplicaciones de vídeo y audio, y en este sentido *JMF* puede jugar un papel esencial, ya que aísla al desarrollador de la mayor parte de los detalles farragosos de este tipo de tecnologías mediante un *API* simple y directo.

NÚMEROS ATRASADOS



159 - Abril 2008

Paseo por The Evolution Show organizado por Microsoft para presentar sus últimos productos. Solución a problemas habituales desarrollando en J2ME pero que pueden mitigarse con J2ME Polish. Cómo generar un entorno de aprendizaje virtual a través del cual alumnos y profesores puedan interaccionar. Nuevas extensiones de Microsoft para ejecución paralela en .NET. Tratamos el Desarrollo de aplicaciones para redes sociales con OpenSocial de Google. Analizamos Java Media Framework (JMF) un software que permite crear aplicaciones Java. 1 CD incluido.



158 - Marzo 2008

Desvelamos las principales novedades de Visual Basic 2008. Análisis sobre la versión final del Service Pack 1 de Windows Vista. Análisis de Microsoft Visual Studio 2008, con la edición Express. Profundizaremos algo más en los tipos de aplicaciones que pueden hacerse con Android y las APIs disponibles.

Descripción de PyS60 versión de Python ideada para dispositivos móviles con sistema operativo Symbian y de tipo Serie 60. Análisis de LINQ para SQL. Creación y gestión de componentes, así como la publicación de nuestras bases de datos en la web. Ponemos en práctica mediante la framework Jena las tecnologías semánticas revisadas en la primera parte. 1 DVD incluido.

157 - Febrero 2008

Nos sumergimos en el mundo Android, la nueva plataforma móvil de la Open Handset Alliance, para analizar las herramientas incluidas en su SDK y poder crear nuestra primera aplicación Android. Además, continuamos nuestro seguimiento a la tecnología 4D v11 SQL, iniciamos una serie dedicada a LINQ para SQL, continuamos nuestro desarrollo para iPhone, programamos la web semántica con Jena y ofrecemos la segunda entrega de nuestro curso AJAX. 1 CD-ROM incluido.



156 - Enero 2008



El iPhone ha irrumpido con fuerza en el mercado. En este número desvelamos las principales técnicas para programar aplicaciones web adaptadas a este innovador dispositivo. Además, entregamos la última parte del curso dedicado a la creación de buscadores con Lucene, una extensa revisión a la tecnología LINQ para XML, hacemos una primera introducción a Silverlight. Además, empezamos una serie avanzada sobre programación AJAX, en la cual crearemos un slideshow de imágenes. En términos de bases de datos, empezamos un tutorial sobre 4D v11 SQL, además de las secciones habituales. 1 DVD incluido.

155 - Diciembre 2007

LINQ, una de las grandes novedades de la nueva generación de tecnologías de Microsoft, permite al desarrollador C# o VB realizar consultas de forma nativa sobre cualquier colección de datos. Empezaremos aquí un extenso análisis de esta tecnología, que nos llevará a cubrir LINQ para objetos, XLINQ y DLINQ. Además, finalizamos nuestro repaso a las novedades de Windows Server 2008, hacemos un resumen de lo mejor del TechEd 2007, entregamos la segunda parte del tutorial dedicado a JavaFX, nos adentramos en el mundo de los buscadores con Lucene, y por último mostramos las capacidades de desarrollo modular que ofrece 4D v11 SQL. 1 CD-ROM incluido.

Si te falta algún número de la temporada, ahora tienes la oportunidad de conseguirlo

Precio Oferta descuento

Precio por ejemplar: 6€

1 a 10 = 10% dto. / 11 a 20 = 20% dto.
21 a 30 = 30% dto. / 31 a 40 = 40% dto.
+40 = 50%

BOLETÍN DE PEDIDO

Rellene o fotocopie el cupón y envíelo a **REVISTAS PROFESIONALES, S.L.**
(Revista **SÓLO PROGRAMADORES**) C/ Valentín Beato, 42 - 3ª Planta - 28037 MADRID
Tel.: 91 304 87 64 - Fax: 91 327 13 03 - www.revistasprofesionales.com - rpsuscripciones@revistasprofesionales.com

Deseo me envíen los número/s: . . . .
NOMBRE Y APELLIDOS . . . . .EDAD . . . . TELÉFONO . . . .
DOMICILIO . . . . .C.P.: . . . .
CIUDAD . . . . .PROVINCIA . . . .

FORMAS DE PAGO

- ❑ Giro Postal a nombre de REVISTAS PROFESIONALES, S.L.
- ❑ Talon Bancario a nombre de REVISTAS PROFESIONALES S.L.
- ❑ Domiciliación Bancaria
- ❑ Contra Reembolso (5€ de gastos de envio por paquete)

Banco . . . .
Domicilio . . . . Firma:
Numero de cuenta: _ _ _ _/_ _ _ _/_ _/_ _ _ _ _ _ _ _ _ _ _
Titular . . . .

- ❑ Tarjeta de crédito _ _ _ _/_ _ _ _/_ _ _ _/_ _ _ _/ Fecha de caducidad:

Extranjero: Gastos de envio 5€ por paquete. Unica forma de pago tarjeta de crédito (VISA, Mastercard, American Express,...)



Preguntas y respuestas

ADOLFO ALADRO GARCÍA

Estoy trabajando en una página *Web* con un *textarea*. Cuando el usuario selecciona un trozo de texto dentro de ese *textarea*, quiero saber en qué posición comienza y termina la selección. Necesito una solución que funcione en *Internet Explorer* y *Mozilla*.

La respuesta más rápida se puede dar en el caso de que el navegador sea *Mozilla* ya que ekl propio objeto correspondiente al elemento *textarea* cuenta precisamente con las propiedades *selectionStart* y *selectionEnd*. No obstante se puede hacer un pequeño desarrollo con el fin de obtener un objeto, válido para los dos navegadores, con el que conocer el comienzo y el final de la selección del usuario.

El objeto *Javascript* se llama *Selection Handler*:

```
function SelectionHandler(oTextarea) {
  this.oTextarea = oTextarea;
  this.oSelection = {start: -1, end: -1};
}
```

El parámetro que recibe es el objeto correspondiente al *textarea*. El constructor del objeto crea primeramente un atributo, *oTextarea*, y le asigna dicho parámetro. A continuación crea un objeto, *oSelection*, con dos atributos, *start* y *end*, inicialmente con los valores -1. Cada vez que se quiera conocer la selección del usuario *SelectionHandler* actualiza los valores de su propiedade *oSelecion* y devuelve dicho objeto. Los métodos del objeto se definen usando la palabra reservada *prototype*:

```
SelectionHandler.prototype = {
  get: function() {···},
  _ie_get: function() {···},
  _mozilla_get: function() {···}
}
```

El método *get* es el que se usa con independencia del navegador. Se encarga de detectar en qué caso se encuentra validando la existencia de la propiedad *selection* del objeto *document* así como la existencia de la propiedad *selectionStart* del objeto correspondiente al *textarea*:

```
if (document.selection != null && this.
oTextarea.selectionStart == null) {
  return this._ie_get();
} else {
  return this._mozilla_get();
}
```

Dependiendo de los resultados obtenidos en las validaciones detalladas se llama al método *_ie_get* o *_mozilla_get* que son los que realmente llevan a cabo la tarea.

Tal y como se ha adelantado el caso más sencillo es el que presenta *Mozilla* ya que el objeto correspondiente al elemento *textarea* tiene dos propiedades que proporcionan directamente los valores buscados:

```
_mozilla_get: function() {
  this.oSelection.start = this.oTextarea.
  selectionStart;
  this.oSelection.end = this.oTextarea.
  selectionEnd;
  return this.oSelection;
}
```

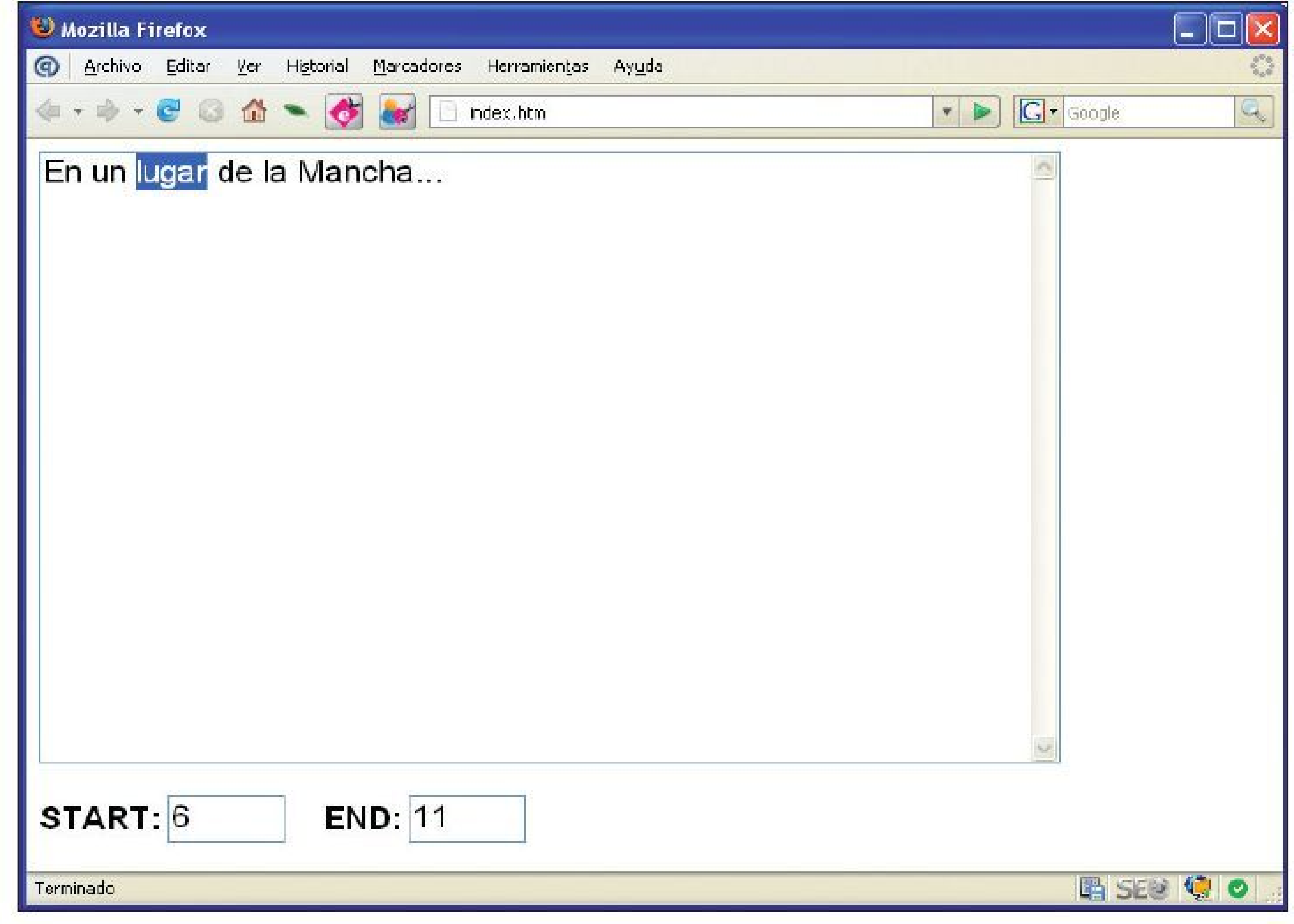
Si el navegador es *Internet Explorer*, la solución es un poco más complicada. A grandes rasgos los pasos a seguir pueden resumirse en cuatro. El primero consiste en crear un marcador. En el segundo paso el marcador se pondrá por delante y por detrás del texto seleccionado. El tercer paso consiste en determinar las posiciones de la selección obteniendo las posiciones del marcador. Finalmente en el cuatro paso se restaura el valor original.

El elemento *textarea* debe coger el foco para que posteriormente se pueda utilizar el método *createRange* sobre la selección actual. Asimismo será preciso obtener un *bookmark* que se empleará posteriormente para dejar todo como estaba:

```
this.oTextarea.focus();

var oRange = document.selection.
createRange();
var oBookmark = oRange.getBookmark();
```

El texto original del elemento *textarea* se almacena en una variable. Seguidamente se crea una cadena de texto que hará las veces de marcador. Hay que garantizar de alguna



Ejemplo de obtención de las posiciones de comienzo y final de una selección dentro de un elemento textarea.



forma que dicho marcador no es una cadena de texto que pueda formar parte del propio valor del elemento *textarera*:

```
var sTxt = this.oTextarea.value;
var sOriginalTxt = sTxt;
var sMarker = "#" + Math.round(1000000*
Math.random()).toString(34) + ' ' +
new Date().getTime().toString(34) + "#";
```

La propiedad *text* del objeto devuelto por *createRange* se actualiza empleando el marcador:

```
oRange.text = sMarker + oRange.text +
sMarker;
sTxt = this.oTextarea.value;
```

Conocer las posiciones de comienzo y final de la selección es tan sencillo como determinar las posiciones de los marcadores. Para ello la propiedad *oSelection* del objeto *SelectionHandler* actualiza sus atributos *end* y *start*:

```
this.oSelection.start = sTxt.indexOf
(sMarker);
sTxt = sTxt.replace(sMarker, "");
this.oSelection.end = sTxt.indexOf
(sMarker);
```

Finalmente el elemento *textarea* debe dejarse tal y como estaba antes de comenzar la operación. Para ello se actualiza el valor del atributo *value* con la cadena de texto original y se ejecuta el método *moveTo Bookmark*:

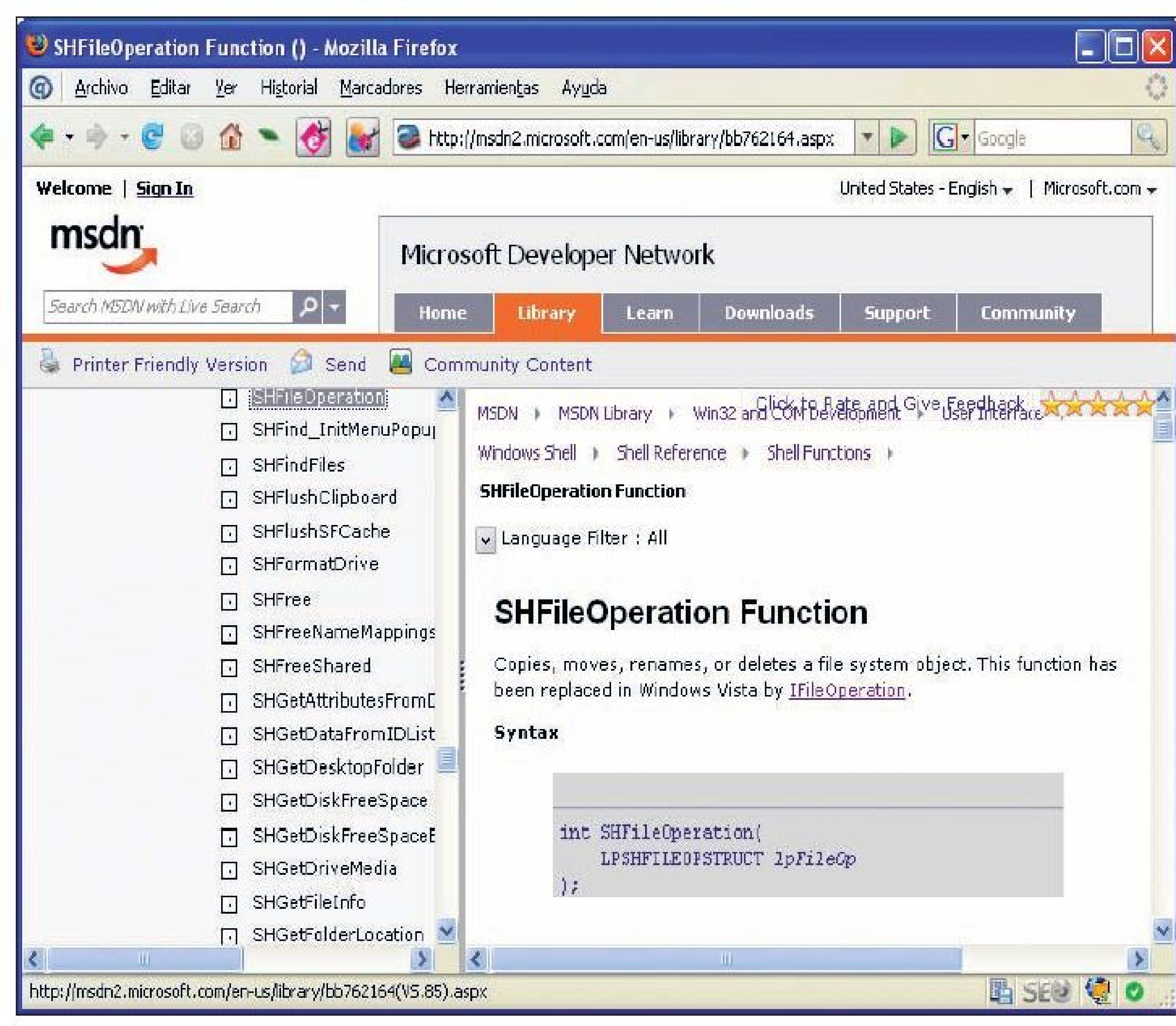
```
this.oTextarea.value = sOriginalTxt;
oRange.moveToBookmark(oBookmark);
oRange.select();
```

El último paso consiste en devolver el objeto *oSelection*:

```
return this.oSelection;
```

En una aplicación en *Windows* programada en *C++* quiero borrar un directorio con todo su contenido, ya sean ficheros o subdirectorios. Con el procedimiento *RemoveDirectory* no es posible. ¿Cómo puedo hacerlo?

La respuesta está en el procedimiento *SHFileOperation* del *API Shell* (*msdn2.microsoft.com/en-us/library/bb762164.aspx*). Sirve para copiar, mover, renombrar o borrar ficheros (En *Windows Vista* ha dio reemplazado por la interfaz *IFileOperation*). *SHFileOperation* se define de la siguiente forma:



Documento del API estándar de Windows sobre el procedimiento SHFileOperation.

```
int SHFileOperation(
  LPSHFILEOPSTRUCT lpFileOp
);
```

El parámetro que recibe es un puntero a una estructura *SHFILEOPSTRUCT*. Ésta debe contener toda la información necesaria para que se pueda llevar a cabo la operación. El procedimiento devuelve cero si ha concluido con éxito o cualquier otro valor distinto de cero en otro caso.
La estructura *SHFILEOPSTRUCT* se define tal y como sigue:

```
typedef struct _SHFILEOPSTRUCT {
    HWND hwnd;
    UINT wFunc;
    LPCTSTR pFrom;
    LPCTSTR pTo;
    FILEOP_FLAGS fFlags;
    BOOL fAnyOperationsAborted;
    LPVOID hNameMappings;
    LPCTSTR lpszProgressTitle;
} SHFILEOPSTRUCT, *LPSHFILEOPSTRUCT;
```

El atributo *hwnd* es el manejador de la ventana que recibirá las respuestas del procedimiento. El atributo *wFunc* identifica la operación. Así por ejemplo su valor tiene que ser FO_DELETE para el caso del borrado. La cadena de texto correspondiente a la ruta del disco que va a ser objeto de la operación. En el siguiente ejemplo, siguiendo con el problema del borrado recursivo, se estaría preparando la llamada para borrar todos los contenidos del directorio *C:\temp* incluyendo el propio directorio *C:\temp*:

```
SHFILEOPSTRUCT sh;
...
sh.pFrom = "c:\\temp\0";
...
```

El atributo *pTo* es similar al anterior. En el ejemplo que nos ocupa no es necesario y debe establecerse a *NULL*:

```
sh.pTo = NULL;
```

El atributo *fFlags* controla los detalles de la operación. Hay muchas opciones. En el ejemplo que se está viendo las opciones elegidas son las que se muestran seguidamente:

```
sh.fFlags = FOF_NOCONFIRMATION |
FOF_SILENT;
```

El *flag FOF_NOCONFIRMATION* indica al sistema que no se desea que se muestre una ventana de confirmación por cada borrado. El *flag FOF_SILENT* indica que no es necesario mostrar una ventana de progreso de la tarea.



Disco Duro Remoto

JORGE RUBIRA

En esta ocasión, vamos a desarrollar una página Web en PHP en la que diseñaremos un disco duro remoto al que podremos crear directorios y subir y bajar ficheros. Para ello, utilizaremos XAMPP, Eclipse y un Plugin PHP Eclipse.

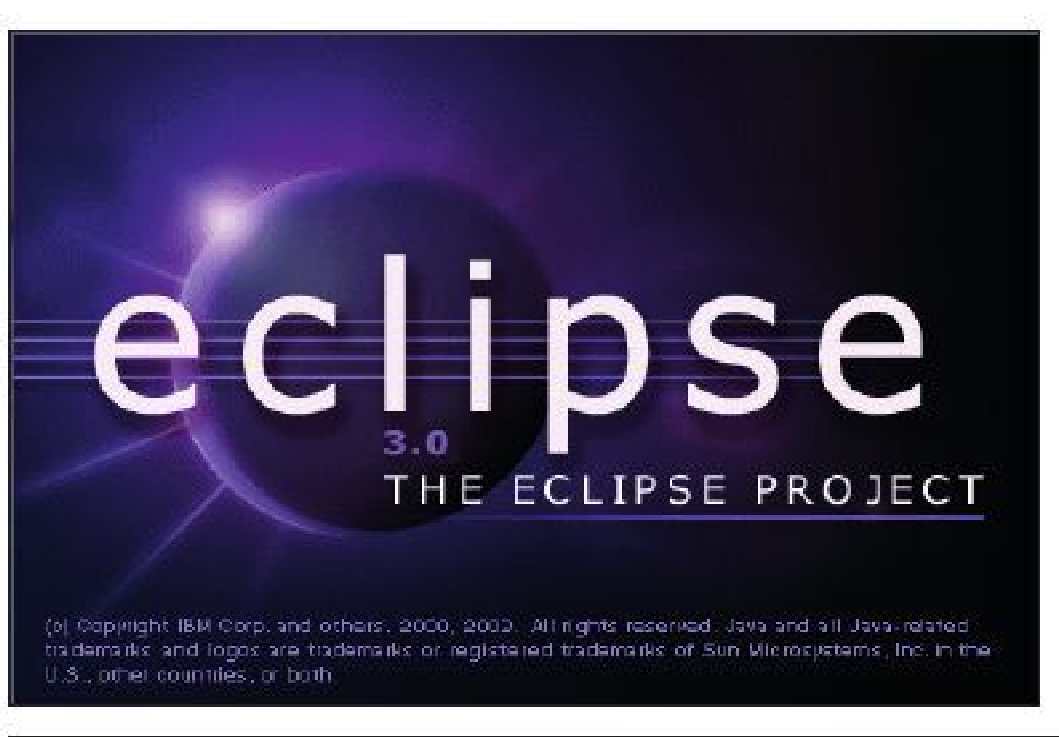
Paso a paso

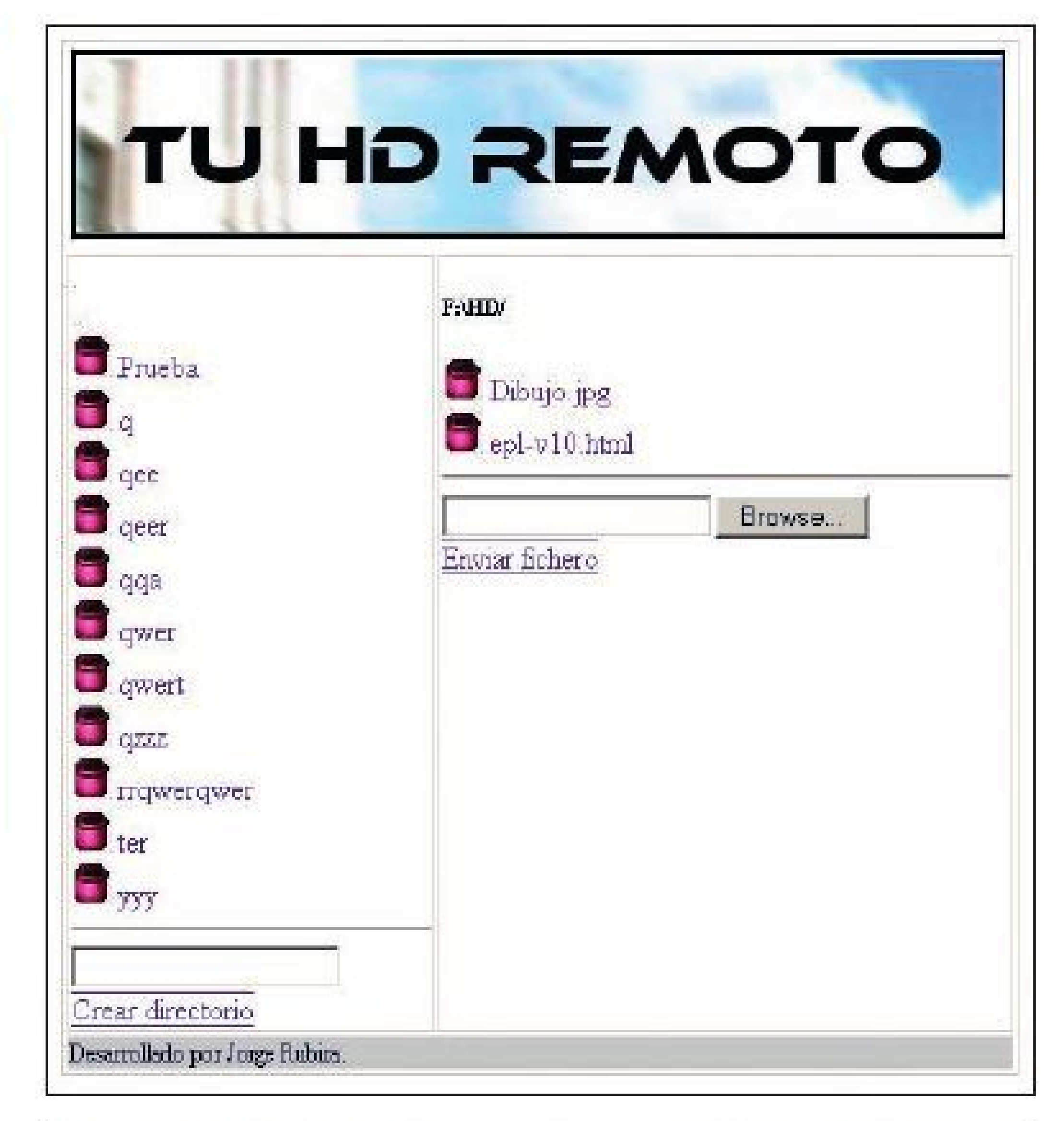
Jorge nos enseñará paso a paso cómo cargar, configurar y utilizar las distintas aplicaciones, y el paso a paso de creación y depuración del disco remoto.

¿Cómo obtener el vídeo-tutorial?

El material que conforma este vídeo-tutorial consiste en un archivo de vídeo y unos archivos de código que implementan el proyecto. Los lectores de la edición en papel encontrarán el vídeo-tutorial en el CD-ROM, mientras que los lectores de la edición digital lo encontrarán en el paquete descargado. Recordamos a los lectores de la edición digital que la nueva dirección es http://www.revistasprofesionales.com

El formato del vídeo es WMV y el tamaño de la descarga es de 50 MB aproximadamente.







¿Tienes sugerencias para el próximo vídeo-tutorial?

Nos interesa saber cómo podemos mejorar los vídeos y sus contenidos, de modo que si tenéis cualquier sugerencia para futuros vídeos, no dudéis en transmitirla a Jorge Rubira, el autor: encuestavideos@gmail.com.



WWW.AGPMCOMPUTERS.ES

AGPM COMPUTERS

AGPM

PORQUE NO TODO ES INFORMATICA...

CIRCUITO DEL JARAMA

ENTRADA GRATUITA

TROFEO RACE

12 ABRIL
24 MAYO
21 JUNIO
13 SEPT

TODA LA INFORMACION EN:

ochocerotres.com

BUSCANOS POR LOS BOXES

Hace poco os contamos lo bien que lo pasamos **en 2007 a lomos de una Yamaha R1** con la pericia de nuestro piloto José Benavente. Este año **2008 repetimos**, esta vez a lomos de una **Suzuki GSXR** y **nos gustaría que nos acompañaseis** este año a disfrutar de nuestro pequeño sueño.

Somos una **tienda de hardware** informático. Estamos especializados en **montaje, instalación y mantenimiento de equipos y redes informáticas.** Trabajamos con todo tipo de **equipos, componentes y periféricos.** Nuestra actividad va desde el **cliente particular** a **empresas** e **instituciones** de diversos campos. En **AGPM Computers** siempre hemos luchado por mantener nuestros **precios competitivos**, sin perder jamás el **trato humano** y la **confianza.**

Un pequeño ejemplo son los **5 años** que llevamos con nuestra **oferta PLAN RENOVE, trae tu vieja CPU** –funcionando- y **te llevas otra nueva** con una configuración competitiva y actual. Todo ello por **270 euros***

(*precio orientativo, del mes de abril de 2008, para más información sobre la oferta visite nuestra pagina web www.agpmcomputers.es)

PLAN RENOVE 

Chasis ATX USBs delanteros + **fuente 420W**
Placa Base **GIGABYTE GA-MA69VM-S2**
Micro AMD 64 bits **LE-1640** AM2 2600 MHz/1Mb
Memoria DDR II **2048 Mb PC800**
Disco Duro **320 Gb SATA2**
Lector frontal de tarjetas digitales 3,5"
Regrabador **DVDROM Dual (+R/-R) LG H58N 20x Doble Capa**
Grafica Integrada
Tarjeta de Sonido Integrada

270€

IVA INCLUIDO

*configuración y precio sujetos a posibles cambios según mercado, consultar web o llamar para mas información

AGPM COMPUTERS

agpm@agpmcomputers.es

AGPM COMPUTERS, tu tienda de informática

No dude en visitarnos en nuestra página web

www.agpmcomputers.es

o ponerse en contacto con nosotros en el **912298497**

Contenido del CD-ROM

Fuentes

Java Media Framework (II)

A lo largo de estos dos artículos se ha hecho una introducción muy somera a la tecnología *Java Media Framework*, especialmente en lo que tiene que ver con la reproducción de audio y vídeo. En realidad detrás de las siglas *JMF* hay mucho más: grabación y reproducción utilizando todo tipo de dispositivos, conversión entre formatos, etc. Parece claro que el futuro inmediato pasa necesariamente por las aplicaciones de vídeo y audio, y en este sentido *JMF* puede jugar un papel esencial, ya que aísla al desarrollador de la mayor parte de los detalles farragosos de este tipo de tecnologías mediante un *API* simple y directo.

Jena (III)

En esta tercera entrega exploramos las tecnologías que constituyen lo que se denomina como "web semántica con minúsculas", es decir, una simplificación práctica de los conceptos y tecnologías de la Web Semántica aplicable desde ya mismo para añadir más inteligencia a nuestros portales Web 2.0.
Este artículo tiene por objeto analizar el conjunto de tecnologías que hacen posible la "web semántica con minúsculas", esto es, cómo acercarnos progresivamente y de una manera más sencilla hacia la todavía quimera de la Web Semántica modificando de manera incremental a través de tecnologías no intrusivas el contenido de marcado de los portales Web 2.0 actuales.

Android (III)

Android es una plataforma incipiente, pero ya en un estado de desarrollo muy avanzado y con un futuro muy prometedor. Apoyada por una compañía como Google, promete dar mucho de qué hablar en los próximos meses.
Desde esta serie de 3 artículos, hemos visto cómo crear una aplicación sencilla desde el principio, y la hemos ido complicando un poco más hasta llegar a temas algo más avanzados. Para poder tener una visión más general, en este número se incluye un pequeño ejemplo realizado en Eclipse que sintetiza todo lo visto hasta ahora. Es muy recomendable revisar su código para completar los ejemplos vistos en los listados. Una vez entendido el ejemplo, ya seremos unos iniciados en la tecnología.



Podcast javaHispano

010 - OpenXava framework (Entrevista a Javier Paniza)

Número décimo de Javahispano Podcast. Este número está dedicado a OpenXava donde Javier Paniza nos hablará de como utilizar dicho framework. Las secciones de este podcast son:

- Sección noticias: Presentada por Abraham Otero y Alfredo Casado
- Mapa de usuarios de javahispano en Google maps
- OpenXava 3.0
- Sun contrata al líder del proyecto Jython (discusión sobre el proyecto Davinci)
- ¿Por qué no hay más JUGs en España?
- Java EE 5 ¡¿la solución más empleada para entornos de producción?!



- Polémica respecto al perfil web para Java EE 6
- Publicado el test Acid 3
- ¿Siguen siendo las bases de datos relacional es la mejor opción?

- Entrevista realizada a Javier Paniza por Erick Camacho y Jorge Rubira. En dicha entrevista Javier nos explicará como funciona OpenXava y para que casos podemos utilizarlo.

Vídeo – Tutorial

En esta ocasión, vamos a desarrollar una página Web en PHP en la que diseñaremos un disco duro remoto al que podremos crear directorios y subir y bajar ficheros.
Para ello, utilizaremos XAMPP, Eclipse y un Plugin PHp Eclipse.
Jorge nos enseñará paso a paso cómo cargar, configurar y utilizar las distintas aplicaciones, y el paso a paso de creación y depuración del disco remoto.

Además ...

Solo Programadores número 158 en formato pdf.